EU/G7 Healthcards - WG7

Interoperability of Healthcard Systems

Part 3

Interoperability Specification

Prepared by

Dr David Markwell, MB BS.

(as coordinator of EU/G7 Healthcards WG7 Interoperability)

ADDRESS FOR COMMENTS
The Clinical Information Consultancy,
93 Wantage Road, Reading, Berkshire, RG30 2SN, UK
Tel +44-118-9584954 Fax +44-118-9504464
Email david@clinical-info.co.uk
Web site: http://clinical-info.co.uk/euhci.htm

Status	: Final Draft with G7 & CTM revisions
Version	: 1.1
Date	: 7 November 1996
Reference	: EUHCIF-3.DOC

STATUS

SUBJECT TO ACCEPTANCE OF LATEST REVISIONS, this version replaces 1.0e FOR IMPLEMENTATION in demonstration of interoperability between CardLink and DiabCard.

DEVELOPED by the Core Technical Group of the EU Healthcards Interoperability Feasibility Study.

AGREED by the EU Healthcards Interoperability Feasibility Study Consensus Meeting on 3 July 1996.

REVISED to take account of continuing discussions of Card Terminal Manager API proposals between MCT and GIP-CPS experts.

REVISED to improve alignment with G7 healthcard interoperability proposals.

NO FURTHER SIGNIFICANT CHANGES are planned until the outcome of the interoperability demonstration becomes clear. However, minor revisions may be issued during the development phase to correct reported errors or inconsistencies and to add clarification.

Table of contents

Tables

Revisions between draft (1.0e and 1.1)

The following changes have been made to align this document with the Card Terminal Manager proposals developed during cooperation between French GIP-CPS experts and German MCT experts.

1. References to the Card Terminal API (CTM-API) changed to refer to the Card Terminal Manager API (CTM-API). The Card Terminal interface specific to a particular Card Terminal is within the Card Interface Environment and is not defined in this document.
2. Chapter 5 discussion of Card Terminal services and the MCT implementation replaced by the relevant subset of CTM-API functions (as in the G7 specification).

The following changes have been made to align this document with current G7 proposals developed in cooperation with Japanese experts.

3. Addition of HciInitialise and HciTerminate functions to make the loading/configuration and unloading of a particular Healthcard Server explicit as part of the healthcard recognition process driven by the Healthcard Client Application or an intermediate Healthcard Server Manager.
4. Simplification of Healthcard Server functions taking account of the configuration performed in the HciInitialise function and reordering parameters to follow the same conventions as those used in the CTM-API and in the G7 Healthcard Server Manager-API.
5. Alignment of HS-API return codes with those that are relevant as specified for the CTM-API.

The following changes have been made to correct and clarify the information in the specification..

6. Addition of descriptions of healthcard interoperability architecture components to allow this specification to stand alone – without the need to refer to Part 1 and Part 2 of the feasibility study report.
7. Revision of the tabular scenario descriptions in earlier versions to follows the calling sequence approach used in the G7 healthcard interoperability specification.
8. More consistent presentation of functions and parameter types.
9. Miscellaneous technical information on parameter types, return codes and configuration information grouped together in a new Section 7 to simplify referencing.
10. Various minor typographical corrections.

Revisions between draft (1.0d and 1.0e)

These changes comprise only include corrections in response to comments and addition of clarification in areas requested.

1. Inconsistencies in format of country codes removed. All country codes in the tabular representation are now length "3 fixed". This applies to Table 7 Country of Birth, Address Country, Contact Address Country, Insuring Body Country, Insuring Body Address Country, Insured Person Address Country and to Table 8 instances in of Author Country in various contexts and Responsible Party Country. This aligns with the format of the Country Code used in other objects such as Card Issuer and Issuer of Patient Identification and in ENV12018. Corresponding changes to ASN.1 definitions have been made so that all relevant items are now of type "NUMERIC STRING SIZE(3)"
2. Removal of inconsistent sub-object capitalisation in 8.4.2 CardApplicationIdentification becomes cardApplicationIdentification.
3. Clarifying note added regarding the representation of sets and sequences using ASN.1 Basic Encoding Rules in 8.2.5

Revisions between draft (1.0c and 1.0d)

These changes include revisions to support the Card Terminal API and corrections of inconsistencies in the specification of data content and

Changes to reflect Card Terminal API specification

1 Change in earlier version to text removed in version 1.1.
2 Change in earlier version to text removed in version 1.
3 Change in earlier version to text removed in version 1.

4. Introduction of a Card Terminal Configuration File. This maps a logical reference to a card (or card connection) at the Healthcard Server API to the parameters required by the MCT CTM-API or by a future Card Terminal API. Thus CTM-API specific parameters are no longer required at the Healthcard Server API. This ensures that future changes to the Card Terminal API do not affect the Healthcard Server API.
5. Change in earlier version to text removed in version 1.

Addition of references to continuing Standardisation work

6. Addition of active standardisation work in CEN TC224 WG12 to the Standards referenced in Annex A.
7. Addition of comments on the relevance of the work of CEN TC224 WG12 to data structures on newly designed healthcards in 8.1.4

Corrections

8. Repetitions of "Medication coding structure" were inconsistently shown as 0,9 in the data tables (1.0b page 3-65 and 3-84) but as SET SIZE(0…6) in the ASN.1 definition (1.0b page 3-95). REVISED to 0,6 throughout.
9. Update details show "Author" as optional in data tables but not in ASN.1 definition (1.0b page 3-95). REVISED to OPTIONAL in the ASN.1 definition.
10. ASN.1 definition inconsistent object name "IssuerOfPatientIdentification" CORRECTED to

"IssuerOfPatientIdentifier" (1.0b page 3-93).

Interoperability of Healthcard Systems

Part 3 - Interoperability Specification

Final Draft with G7 & CTM revisions

Preface

This document is the third of three parts of the report of the Healthcard Interoperability Feasibility Study. It specifies an approach to interoperability between EU funded healthcard projects. This specification is based on the general concepts in Part 1 and the more detailed considerations in Part 2.

It is intended that this specification should be used as the basis of the interoperability demonstration between CardLink and DiabCard. Readers of this document should note that, while there are differences between this specification and the technical formats used in CardLink 1 and DiabCard 2, these changes do not alter the ability to convey the minimum data sets specified by those projects. Furthermore, with appropriate interface software, the interoperable administrative and emergency data on existing cards should be readable using the API specified in this document. The principal reason for these changes is to provide a structure that is readily able to accommodate the evolution of future requirements while supporting the use of minimal emergency cards issued by low cost systems. Extensions and variations from the CardLink specification have taken account of the appropriate European and International Standards and in particular ENV12018.

The current draft has been developed by the EU Healthcard Feasibility Study - Core Technical Group (CTG) during the first half of 1996. A consensus meeting to which all EU funded healthcard projects were invited was held on 3 July 1996. At that meeting the specification was accepted, subject to a number of specified amendments. One area of continuing uncertainty in earlier drafts was the Card Terminal Manager API. This was the subject of continuing discussions between French and German experts. These discussions have now reached a satisfactory consensus and the specification has been revised to take account of this.

This version is to be considered by those responsible for developing interoperability demonstrations between DiabCard and CardLink. It was previously announced that there would be no significant changes until the outcome of the interoperability demonstration became clear. However, the revisions in this document are expected to substantially increase the value and use of products developed by the projects. Therefore, those implementing the demonstrators are recommended to accept these revisions. Future, minor revisions may be issued during the development phase to:

- Correct reported errors;
- Remove ambiguities or inconsistencies between different parts of the document;
- Add clarification;

No other changes will be made unless unanimously agreed by those responsible for development.

The parts of the document are available in Microsoft Word™ 6.0 files. These files are named in accordance with the following convention – – EUHCI*[a]-[n]*.DOC

[a] is a letter signifying the version of the document (NB "F" = version 1.1)
[n] is a digit signifying the part of the document.

Note: Only Part 3 has been changed in the latest corrected issue.

1 Healthcard System modules and interfaces

1.1 Overview

For the purposes of this specification, we define four components and three distinct interfaces at which interoperability is required. These are illustrated in Figure 1 and described on the following pages.

The main interoperability requirements are specified at one of the interfaces shown in this diagram. Healthcard Server API (Chapter 4), Card Terminal Manager API (Chapter 5) and Card Interface (Chapter 6). Data content and structure are relevant to all layers and are addressed separately in Chapter 8.

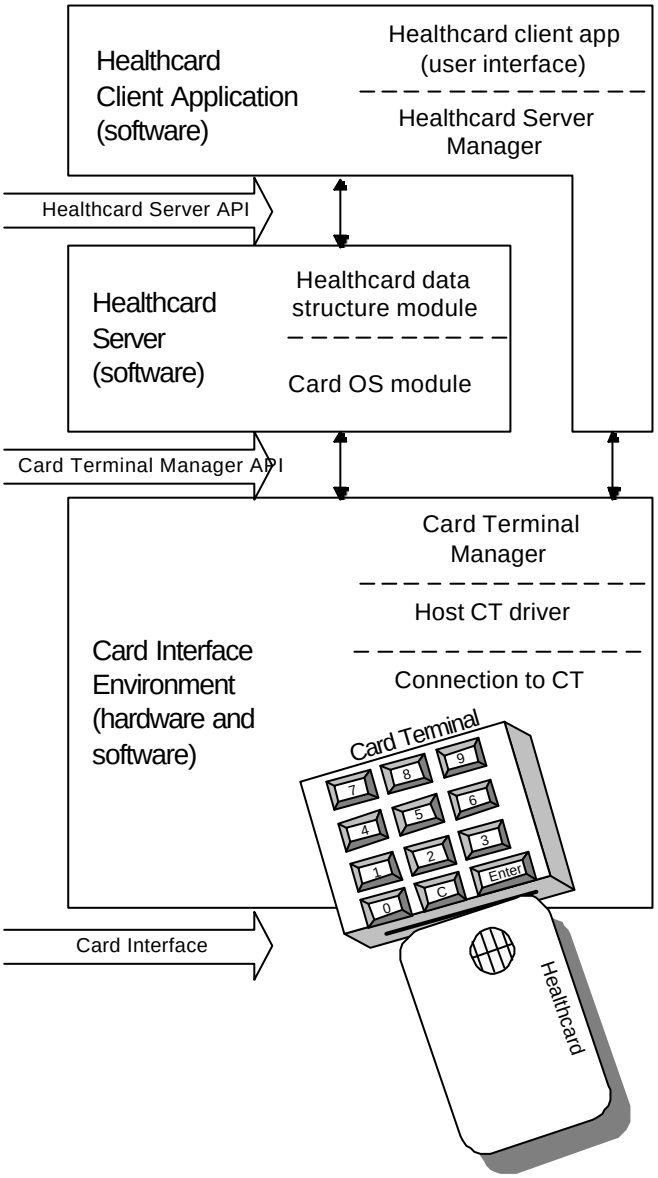

Figure 1. A modular approach to Healthcard System interoperability

1.2 Components

1.2.1 Healthcard Client Application

A Healthcard Client Application consists of software which provides a user interface to a healthcard system. A Healthcard Client Application accesses interoperable healthcards using services provided by Healthcard Servers and a Card Terminal Manager. These services are accessed through two interfaces – the Healthcard Server API (HS-API) and the Card Terminal Manager API (CTM-API).

Within the EU Healthcard Interoperability architecture, the Healthcard Client Application uses CTM-API functions to recognise the type of card that has been inserted. It then loads and/or configures an appropriate Healthcard Server. It then uses the Healthcard Server API to read data from interoperable healthcards.

The role of recognising cards and loading appropriate Healthcard Servers can be separated from the Healthcard Client Application in a layer referred to as the Healthcard Server Manager. This separation is optional in EU systems but is required for wider interoperability at the G7 level. The role of the Healthcard Server Manager is formalised in the G7 Healthcard Interoperability specification which includes a definition of a Healthcard Server Manager API (HSM-API) that mirrors the functionality of the EU HS-API.

1.2.2 Healthcard Servers

Healthcard Servers are software modules that provide access to data on different healthcards through a common interface – the HS-API. They must read and map the data stored on the card so that it is presented in a common format at the HS-API. They must use Card Terminal Manager API functions to access the card and to control the Card Terminal.

Different Healthcard Servers may be used for each healthcard with a different data format. Alternatively some Healthcard Servers may be configurable to support several different healthcard formats. The development of individual Healthcard Servers is the responsibility of healthcard designers.

A Healthcard Server may have two distinct sub-components with different roles:

- The *healthcard data structure module (HDSM):*
  - Receives HS-API commands and maps these to requests to read card data;
  - Receives data from the card and maps it to the form required for the HS-API.
- The *card operating system module (COSM):*
  - Maps the requests for card access from the HDSM into commands that are appropriate to the operating system of the card and submits these commands via the CTM-API.
  - Receives raw data back from the CTM-API and presents this to the HDSM for mapping.

1.2.3 Card Interface Environment

Hardware and software that together provides the Card Terminal Manager API and communicates with one or more cards. The components of the Card Interface Environment include:

- One or more Card Terminals (or reader):
  - Slots for one or more ID-1 IC-cards;
  - Optional contacts for one or more ID-000 plug in IC cards;
  - Optional built in keypad and/or display.
- Card Terminal software/firmware:
  - Located within the Card Terminal.
- Host to Card Terminal connection:
  - Connected via a communication link (e.g. RS232);
  - Integrated with host system (e.g. MCIA, SCSI, etc.).
- Host software:
  - Located in the host environment;
  - Provides the Card Terminal Manager API;
  - Manages communication with the Card Terminal.

Host software may optionally be divided into two distinct parts.

- A proprietary device driver that provides an interface to the Card Terminal.
- A non-proprietary Card Terminal Manager that provides the CTM-API and supports one or more interfaces to Card Terminal device drivers. The German MCT is an example of a possible interface to the device driver.

1.2.4 Healthcard

A computer readable card used in connection with the administration or provision of healthcare. In the context of this specification this is an IC Smart Card that conforms with parts 1 to 4 of ISO 7816.

1.3 Configuration files

Two configuration files are defined in this specification. One of these is used when recognising cards and selecting the appropriate Healthcard Server, the other is used by the Card Terminal Manager. These configuration files are specified in Section 7.2.

1.3.1 Healthcard Server Manager Configuration file

The Healthcard Server Manager Configuration file includes details of the Healthcard Servers available. For each Healthcard Server, data is provided that details the name of the library file and the factors by which appropriate healthcards are recognised (e.g. ATR and Card Application Identifiers).

This file is read by the Healthcard Client Application (or an intermediate Healthcard Server Manager) to provide a template for matching an inserted card against the available Healthcard Servers. If a match is found, the appropriate Healthcard Server is then loaded to allow the card to be read.

Suppliers of healthcards and Healthcard Servers must provide the relevant entries in a form that can be appended to an existing configuration file.

1.3.2 Card Terminal Manager Configuration file

Card Terminal Managers require configuration information including details of the local set up of Card Terminals and the appropriate driver software.

This information associates a logical name with:

- A Card Terminal;
- The host software driver that should be loaded to address that Card Terminal;
- Other information required by driver software on the host to allow it to address a particular slot in a particular Card Terminal. This information will vary according to the capabilities of the Card Terminal and the specification of the driver software. It may include:
  - The physical port or address through which the host is linked to that Card Terminal;
  - The sub-address if more than one Card Terminal is connected to a single port;
  - The functional unit number of a particular slot in that Card Terminal;
  - The functional unit number of the Card Terminal itself for receiving basic commands;
  - The functional unit number of any attached screen or keyboard.

This information is checked by the Card Terminal Manager when it opens a link to a particular slot in a Card Terminal. The relevant information about the specified slot is used to allow the Card Terminal Manager to pass the correct parameters to the CT-API in response to subsequent CTM-API functions.

The Card Terminal Manager configuration file must be updated:

- When installing a new Card Terminal or changing the connection between the host and the Card Terminal.
- When upgrading Card Terminal driver software.
- When installing an application that uses a particular logical name to refer to a resource.

2 Calling sequences

2.1 Introduction

This chapter outlines the calling sequences for each of the layers in the healthcard interoperability architecture. The outlines in this section highlight the points relevant to particular uses. The input and output parameters listed omit those that are related to buffer size and location. Formal definitions of the functions outlined in this chapter are found in the subsequent chapters.

2.2 Healthcard Client Application calling sequence

2.2.1 Open a logical link to Card Terminal Manager API

When an interoperable Healthcard Client Application first loads, it must open a logical link to the Card Terminal Manager API (CTM-API). This function creates a resource table for the calling application and provides a handle which the application must use in when opening Card Terminal resources such as ICC slots.

Function name	CtmOpen	See 5.3.1
Input parameters	None	
Output parameters	AppHdl	Healthcard Client Application stores this for subsequent use to address the CTM-API.

2.2.2 Open a logical link to an ICC slot

When an interoperable Healthcard Client Application first loads, it should open logical links to any Card Terminal resources that it requires. These links must be established through the CTM-API.

For the purposes of interoperability, the Healthcard Client Application must open a logical link to the ICC slot in which interoperable healthcards will be inserted.

Function name	CtmResOpen	See 5.3.3
Input parameters	AppHdl	As stored following CtmOpen (see 2.2.1)
	SzResName	Reference to Card Terminal Manager configuration information for the ICC slot.
Output parameters	ResHdl	Healthcard Client Application stores this for subsequent use to address the resource.

2.2.3 Insert, power-on and reset a card

When the user carries out an activity that requires the presence of a healthcard in the ICC slot, the Healthcard Client Application must call a CTM-API function to request insertion of the card. This function carries out different processes according to the presence and state of a card. If a card is already present and powered-up, its state is unchanged but its ATR (stored by the Card Terminal Manager) is returned. Otherwise, if a card is present in the slot when the function is completed, it is powered-up and reset and the ATR is returned.

Function name	CtmCardOpen	See 5.4.2
Input parameters	ResHdl	As stored following CtmResOpen (see 2.2.2)
	SzPrompt	Optional prompt text for insertion.
	EffectFlags	Optional beep and/or LED prompt for insertion.
	TimeOut	Time in seconds to wait for card insertion.
Output parameters	ATR	Used to check if this is a "native" card directly accessible by the Healthcard Client Application.
	CardState	Card inserted, Card powered-up, Card-changed since last CtmCardOpen for same ResHdl.

On completion: If the ATR shows that this is not a "native" card, continue from 2.2.5.

2.2.4 "Native" card interactions

Once a card is inserted and powered-up, it may be found to be a "native" card of the Healthcard Client Application. In this case, system specific processing may take place (e.g. to read and/or write card data).

This native card processing should access the card – and, if necessary, control the Card Terminal – using CTM-API functions. The CTM-API function that should be used for access to the card is CtmCardCommand (see 5.4.5). When a sequence of context-dependent commands are sent to the card, the context should be protected by using the CtmSetExclusivity function (see 5.3.4) to establish, and subsequently release, exclusive control over the relevant resources.

Support for local card interactions is optional. Some Healthcard Systems may only provide interoperable access and may not support any additional functions with "native" cards.

On completion: If this is a "native" card, continue from 2.2.12 below.

2.2.5 Open G7 healthcard interoperability access

If a Healthcard Server Manager is used, the following steps are replaced by those specified in the G7 Healthcard Interoperability specification. The common calling sequence resumes from 2.2.12

2.2.6 Attempt to recognise the card based on the ATR

The ATR returned by CtmCardOpen (see 2.2.3) is checked against the ATRs expected from cards supported by different Healthcard Servers. This check is performed by reference to a Healthcard Server Manager configuration file that is updated when any Healthcard Server is installed on a system.

2.2.7 Load and initialise the appropriate Healthcard Server

When an interoperability healthcard is to be accessed, the Healthcard Client Application loads and configures the appropriate Healthcard Server. The manner in which this Healthcard Server is selected is operating environment dependent. Once the appropriate Healthcard Server has been loaded, it is initialised. The Healthcard Client Application associates an appropriate Healthcard Server and/or configuration with an ICC slot specified using a CTM-API resource handle (*ResHdl*).

Function name	HciInitialise	See 4.2
Input parameters	ResHdl	As returned by CtmResOpen
	szHsConfig	An optional reference to Healthcard Server configuration information. This is recognised by the Healthcard Server and used to set configuration options. The method of configuration is Healthcard Server dependent. If the Healthcard Server is not configurable, this parameter is ignored.
Output parameters	None	

2.2.8 Attempt to recognise the card based on the Card Application Data

The CardApplicationData, obtained by using the HciReadData and HciGetData functions, is checked against the application identifiers expected from cards supported by different Healthcard Servers. This check is performed by reference to a Healthcard Server Manager configuration file that is updated when any Healthcard Servers are installed on a system. If necessary, a different Healthcard Server is loaded and configured by repeating 2.2.7.

2.2.9 Initiate reading of a healthcard interoperability dataset

When the Healthcard Client Application needs to read an interoperable data set from a healthcard, it must request the Healthcard Server to start reading data from the card.

This function does not return the data read from the card as reading continues asynchronously. The HciGetData functions is used to access data that has been read from a card (see 2.2.10).

Function name	HciReadData	See 4.4
Input parameters	ResHdl	As returned by CtmResOpen
	HciDataSet	Reference to the data set to be read. 0 = CardApplicationData, 1= Administrative Data, 2 = Clinical data
Output parameters	None	

2.2.10 Get healthcard interoperability data read from a card

Once the Healthcard Client Application has requested the Healthcard Server to start reading card data, it should intermittently check for completion of reading. When reading is complete, the data set read from the card is available to the Healthcard Client Application, in the structure defined in this specification. The Healthcard Client Application then processes and/or displays the information for the user.

Until all the specified data has been read from the card and mapped to the required data structures, the HciGetData function returns an error indicating that the reading process is incomplete.

Function name	HciGetData	See 4.5
Input parameters	ResHdl	As returned by CtmResOpen
	HciDataSet	Reference to the data set to be read. 0 = CardApplicationData, 1= Administrative Data, 2 = Clinical data
Output parameters	HciData	Data read from the card formatted in accordance with this specification.

On completion: Optionally, repeat from 2.2.8 for other interoperability datasets.

2.2.11 Terminate the use of a Healthcard Server

When a session of interactions with an interoperable card is complete, the Healthcard Client Application should terminate the Healthcard Server. This allows the Healthcard Server to clear any temporary buffers allocated while processing a card.

Function name	HciTerminate	See 4.3
Input parameters	ResHdl	As returned by CtmResOpen
Output parameters	None	

Important note

To avoid loss of card context, a Healthcard Client Application must not use the resource handle passed to the Healthcard Server in any direct CTM-API function calls while the link through the Healthcard Server is open. Thus, once a particular value of ResHdl has been used in a HciInitialise function, this handle must not be used in any direct CTM-API calls until the HciTerminate function has been used to terminate the HS-API link.

2.2.12 Power-off a card and allow or force card removal

When interaction with an inserted card is completed, the Healthcard Client Application should request the Card Terminal Manager to power-off the card and prepare it for removal. If the card is still in the slot after a specified period of time an error is returned.

Function name	CtmCardClose	See 5.4.3
Input parameters	ResHdl	As stored following CtmResOpen (see 2.2.2)
	SzPrompt	Optional prompt text for removal.
	EffectFlags	Optional physical eject, beep and/or LED prompt for removal.
	TimeOut	Time in seconds to wait for card insertion.
Output parameters	CardState	Card inserted, Card powered-up, Card-changed since last CtmCardOpen for same ResHdl.

On completion: Optionally repeat from 2.2.3 for other cards.

2.2.13 Close the logical link to the Card Terminal Manager

Before terminating, the Healthcard Client Application must close its logical link with the Card Terminal Manager.

Function name	CtmClose	See 5.3.2
Input parameters	AppHdl	As stored following CtmOpen (see 2.2.1)
Output parameters	None	

A separate function (CtmResClose) is specified for closing logical links to individual resources created using CtmResOpen. However, CtmClose implicitly closes all open resources and releases all associated exclusivities. Therefore, CtmResClose need not be used by an interoperable Healthcard Client Application.

2.3 Healthcard Server calling sequences

2.3.1 Response to HciInitialise

In response to the HciInitialise function the Healthcard Server configures itself so that it is ready to read a card. This configuration is associated with a particular resource, specified by the ResHdl passed in the function call. If the Healthcard Server supports several different healthcard formats (or different versions of the same type of healthcard) the Healthcard Server refers to configuration information indicated by the calling parameter *szHsConfig*.

2.3.2 Response to HciReadData

The HciReadData function is asynchronous. Thus, the Healthcard Server returns the code HCI_OK when it accepts a valid request. Processing then continues until all the necessary data has been read and processed.

When responding to an HciReadData function the Healthcard Server should first establish exclusive access to the relevant ICC slot.

Function name	CtmSetExclusivity	See 5.3.4
Input parameters	ResHdl	Passed in HciReadData.
	ExclScope	1 = Exclusive control over this ICC slot.
Output parameters	None	

The Healthcard Server issues a sequence of commands to the card via the Card Terminal Manager. The precise nature of the commands depends on the card operating system and the structure of data on the card. The commands and responses are passed transparently through the CTM-API to the card. The Card Terminal Manager and Card Interface Environment add and subtract protocol data but do not alter the content or structure of the commands or responses.

Function name	CtmCardCommand	See 5.4.5
Input parameters	ResHdl	Passed in HciReadData.
	Command	The ICC command
Output parameters	Response	The ICC response
	CardState	Card inserted, Card powered-up, Card-changed since last CtmCardOpen for same ResHdl.

The Healthcard Server must map the data read from the card to the EU Healthcard Interoperability structure. It is placed in a buffer which can then be accessed by a calling application using the function HciGetData.

When the necessary data has been read, the Healthcard Server releases its exclusive access control over the ICC slot.

Function name	CtmSetExclusivity	See 5.3.4
Input parameters	ResHdl	Passed in HciReadData.
	ExclScope	0 = No exclusive control over this resource.
Output parameters	None	

2.3.3 Response to HciGetData

If the previous HciReadData process is incomplete, the Healthcard Server must return the value HCI_READ_INCOMPLETE.

If the HciDataSet specified in the HciGetData function does not match the same parameter in the previous HciReadData, the Healthcard Server must return the error HCI_DATASET_MISMATCH.

If no error occurs, the Healthcard Server returns the requested EU Healthcard Interoperability data structure, obtained by the previous HciReadData, in the referenced buffer.

2.3.4 Response to HciTerminate

Clear all buffers associated with this link to the Healthcard Server. According to the operating environment, the Healthcard Server may be unloaded or may remain loaded to respond to further requests.

2.4 Card Interface Environment calling sequences

Note that only those functions required for EU healthcard interoperability are outlined here. The actions are presented in the form that they would be undertaken by a Card Terminal Manager acting as the upper-layer of the Card Interface Environment. The actions are therefore described as passing functions to a lower-layer Card Terminal API (CT-API). In the EU interoperability demonstration there may be no clear division between the Card Terminal Manager layer and other host software. In this case card related actions are undertaken directly without passing them to a CT-API.

2.4.1 Response to CtmOpen

Create a resource table for the calling application.

Return a handle that is associated with that resource table (AppHdl).

2.4.2 Response to CtmResOpen

Check configuration information associated with the resource name parameter (SzResName).

Create an entry for the named resource, in the resource table specified by the AppHdl passed by the calling entity. This table entry should contain references to the associated drivers and physical resources (e.g. the port, Card Terminal and slot within a Card Terminal). The exclusivity scope for this resource is initially set to zero.

If necessary, load the relevant CT driver software to provide the CT-API.

Return a handle that refers to that entry in the table (ResHdl).

2.4.3 Response to CtmSetExclusivity

Check that no physical resource in the scope specified by the combination of ResHdl and ExclScope is already within the scope of an exclusivity setting of another calling entity. This can be established by checking entries in other resource tables.

Check that any current or pending operations requested by other applications are not using a physical resource for which exclusivity is requested (this is needed only if asynchronous functions are supported).

Set the requested exclusivity scope in the resource table entry associated with ResHdl.

2.4.4 Response to CtmClose

Clear the resource table associated with the AppHdl passed in the function call.

This implicitly removes any exclusivity settings associated with any resource that were set by this calling entity.

This function does not implicitly power-off the card inserted in an ICC slot associated with resources listed in the resource table that is cleared.

2.4.5 Response to CtmCardOpen

Check that the physical resources associated with ResHdl are not in the exclusivity scope of another application.

Initiate the relevant CT-API functions to check the card state.

If a card is still inserted and powered up:-

Check the card-changed marker for this resource.

If the card has not been changed:-

Return the ATR stored when the card was last reset.

Return the card state indicating that the card has not been changed.

Do not update any "card-changed" markers associated with this ICC slot.

If the card has been changed:-

Return the ATR stored when the card was last reset.

Return the card state indicating that the card has been changed.

Clear the "card-changed" marker for this resource so that future references to this ResHdl return "card not changed".

Do not update any other "card-changed" markers associated with this ICC slot.

If there is no card or the card is not powered-up:-

Initiate the relevant CT-API functions to insert, power-up and reset the card.

(If the CT-API functions do not support a time-out, the Card Terminal Manager should mimic this by polling the CT-API until the card is inserted or the time-out period has elapsed.)

If a card is inserted:-

Store the returned ATR

Set the "card-changed" marker associated with all resource table entries that point to this physical ICC slot.

Return the card state indicating that the card has been changed.

Clear the "card-changed" marker for this resource – so subsequent references to this ResHdl report that the card has not been changed (since this request).

If a card is not inserted within the allowed timeout:-

Return a timeout error and the appropriate card state.

Set the "card-changed" marker associated with all resource table entries that point to this physical ICC slot.

2.4.6 Response to CtmCardCommand

Check that the physical resources associated with ResHdl are not in the exclusivity scope of another application.

Check the "card-changed" marker and combine this with the card insertion and power state to produce the CardState return value.

Pass the command to the card via the CT-API. Receive the response and return it to the calling entity.

2.4.7 Response to CtmCardTest

Check that the physical resources associated with ResHdl are not in the exclusivity scope of another application.

Initiate the relevant lower-level commands to test if the card is inserted and powered-up.

If no card is inserted or the card is not powered up, set the "card-changed" marker associated with all resource table entries that point to this physical ICC slot.

If a card is inserted and powered up, do not change the "card-changed" marker.

Check the "card-changed" marker. Combine this with the card insertion and power state to produce the CardState return value.

2.4.8 Response to CtmCardClose

Check that the physical resources associated with ResHdl are not in the exclusivity scope of another application.

Initiate the relevant lower-level commands to power-off and prompt for card removal.

If the lower-level interface does not support a time-out, the Card Terminal Manager should mimic this by polling the lower layer until the card is removed or the time-out period has elapsed.

Unless the function is rejected by the Card Terminal, set the card-changed marker associated with all resource table entries that point to this physical ICC slot.

Check the card-changed marker. Combine this with the card insertion and power state to produce the CardState return value.

3 Requirements

3.1 User requirements

This chapter summarises the technical requirements for the different components of an interoperable Healthcard System. The starting point is a simple statement of user requirements for interoperability. These can be summarised as follows.

1. The healthcare professional requires a Card Access System that:
   - Automatically recognises interoperable healthcards and adapts to read them;
   - Reads interoperable data from cards issued by other Healthcard Systems;
   - Displays interoperable data in a form that he/she can understand;
   - Is integrated with other operational information systems that he/she uses.
2. The patient requires a healthcard that:
   - Stores relevant interoperable administrative and emergency clinical data;
   - Provides read-only access to appropriate interoperable information through any interoperable Card Access System;
   - Secures confidential information against unauthorised access;
   - Protects the integrity of the information stored against unauthorised updates.
3. The system purchaser requires:
   - Manufacturer independence in respect of cards;
   - Manufacturer independence in respect of card terminals.
4. The system developer and integrator requires:
   - Technology-independence in respect of cards;
   - Technology-independence in respect of card terminals.

Applying these requirements to the model of the Healthcard System presented in Chapter 1 we can divide the system into two parts for which separate requirements can be stated. These parts are the “Local System Components” and the “Card Dependent Components”. They are illustrated in Figure 2. The Local System Components need to meet the requirements under point 1 above, while the Card Dependent Components must meet those under point 2. The requirements under 3 and 4 are dependent on lower level Standards applicable to hardware and operating systems in sets of components.

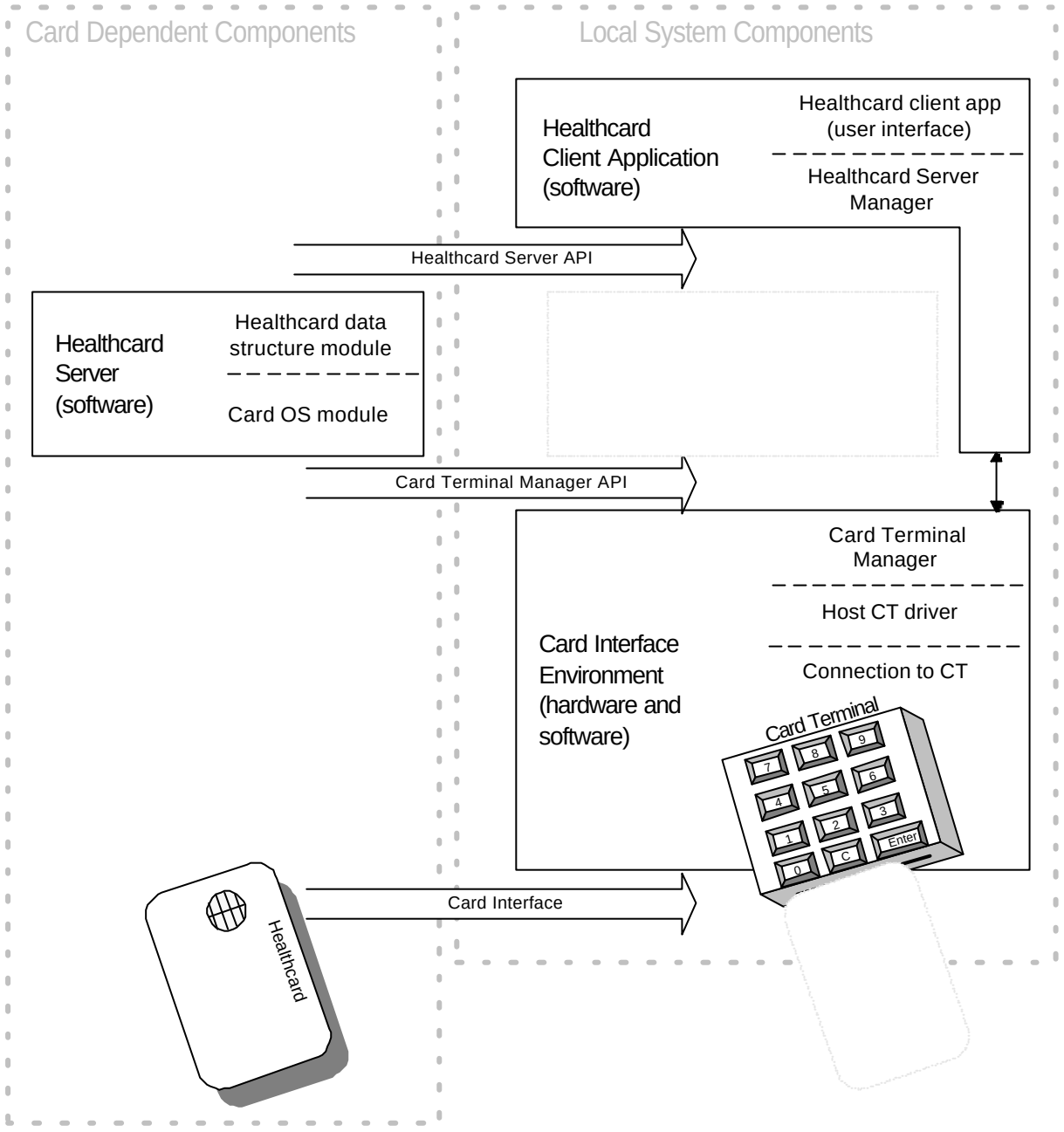

Figure 2. Dividing the system into user and card dependent components

3.2 Requirements for local system components

3.2.1 Client application requirements

An interoperable Healthcard Client Application shall[1]:

- Provide a user interface that enables the user to request the system to read an interoperable healthcard;
- When the user issues such a request:
  - *Interact with a Card Terminal attached to the system (via the CTM-API) to:*
    - *Confirm the presence of a card;*
    - *Power up and reset the card;*
    - *Read the historical bytes of the ATR.*
  - *Check a configuration file that relates card operating systems (denoted by the historical byte of the ATR) to Healthcard Servers and load an appropriate Healthcard Server;*
  - *Recognise the healthcard format by one or more of the following methods:*
    - *Check for an Application Identifier in the ATR (or an ATR file or DIR file if these are accessible);*
    - *Read the Card Application Data from the card using the Healthcard Server API functions described in Chapter 4;*
    - *Try alternative Healthcard Servers appropriate to the card operating system if the loaded Healthcard Server fails when attempting to read Card Application Data from the card.*
  - *Check a configuration file that relates Card Application Identifiers or Card Issuer Identifiers to Healthcard Servers and load the appropriate Healthcard Server.*
  - Read the Administrative and Emergency Clinical data from the card using the Healthcard Server API functions described in Chapter 4.
- Display information read from a healthcard using the Healthcard Server API in such a form that user is able to view at least those items marked as Mandatory (M) in the Application visibility column of Table 13 and Table 14.
  - Ideally the items marked as Recommended (R) should also be visible.
  - Interoperable data items that are not present on a particular interoperable healthcard should be left blank on the display or should indicate that the information is not recorded on the card or is not supported by the card.
  - Interoperable data items that are present but are not accessible due to security restrictions should be displayed in exactly the same way as data items that are not recorded.
- *Enable an interoperable healthcard to be powered off and removed from the Card Terminal.*

[1] Requirements shown in italics may be performed using a separate Healthcard Server Manager module in accordance with G7 healthcard interoperability proposals.

3.2.2 Card Interface Environment requirements

The Card Interface Environment consists of a combination of:

- One or more Card Terminals (readers);
- Optionally, software in the Card Terminal(s);
- A connection to the local host computer system (e.g. serial, bus, SCSI, network);
- Software on the host computer system. This software may optionally be divided into two parts:
  - A proprietary device driver that provides the interface to the Card Terminal.
  - A non-proprietary Card Terminal Manager that provides the CTM-API and supports one or more interfaces to Card Terminal device drivers. The German MCT is an example of a possible interface to the device driver.

The Card Interface Environment:

- Shall provide a Card Terminal Manager API (CTM-API) that conforms with the requirements stated below.
- Shall support communication with T=0 and T=1 cards conforming with ISO7816 by adding and subtracting the protocol in a manner that is invisible to the CTM-API.
- Shall, with the exception of protocol addition and subtraction noted above, enable transparent communication of commands and data from the CTM-API to the card.
- May optionally provide access to synchronous memory cards in the manner described in the German Multifunctional Card Terminal (MCT) specification.

The Card Terminal Manager API provided by the Card Interface Environment:

- Shall provide a Card Terminal Manager API (CTM-API) function calls specified in section Chapter 5.
- May additionally support other Card Terminal Manager API functions.

3.2.3 Note on Card Terminal interoperability

Any Card Terminal Manager API that supports transparent communication with T=0 and T=1 cards is capable of providing access to interoperable healthcards. This document specifies the API functions required to ensure that different Healthcard Servers can access essential Card Terminal services on the different systems to which they are distributed.

The requirements do not ensure that Card Terminals can be interchanged. Card Terminal interoperability is desirable and future developments should therefore conform with a more complete common Card Terminal Manager API specification. The form of such a specification remains under discussion within the remit of the TrustHealth project.

3.3 Requirements for card dependent components

3.3.1 Healthcard requirements

An interoperable healthcard shall:

- Conform to ISO 7816 parts 1 to 4.
- Support access to the Card Application Data described in 8.2.7 in a simple manner that is publicly specified and precisely aligned with the method used by any other healthcards that return the same Historical Bytes in the ATR.
- Allow storage of the data items marked as "Mandatory" (M) in the Card Design Optionality columns of Table 13 and Table 14.
  - It should also allow storage of as many of the items marked as "Recommended" (R) as possible taking account of the card capacity and other design constraints.
- Enable read-only access to the interoperable data set without the use of a specific security devices or procedures.
  - Information that the author or patient wishes to restrict from free-reading may be excluded from the interoperable data set or may be made accessible by use of an interoperable security device or procedure (to be defined by a current or future healthcare professional interoperability initiative).
- Prevent unauthorised updating of the interoperable healthcard data.

3.3.2 Healthcard Server requirements

Issuers of interoperable healthcards shall:

- Provide interoperable Healthcard Server software that enables access to the healthcards they issue in accordance with the requirements stated below.

Interoperable Healthcard Server software:

- Shall be provided in a form that will run under Windows™ 3.1 (and later Windows™ versions);
- Should ideally also be developed for other commonly used operating environments;
- Should be made widely available either free or at a nominal distribution charge for use in any interoperable Healthcard System (this will involve reciprocal exchange of Healthcard Server modules between interoperable healthcard system providers);
- Shall be distributed with a configuration file that facilitates loading of the Healthcard Server when a card to which it relates is inserted. This file shall be formatted as specified in 1.3.1.

When loaded, the Healthcard Server shall:

- Respond to the Healthcard Server API function calls specified in Chapter 4.
- Communicate with the healthcard through the CTM-API using the CtmCardCommand function (specified in 5.4.5) to obtain the relevant interoperable data.
- Return interoperable data read from the card to the Healthcard Server API in the structures and representations specified in Chapter 8.
  - The data returned shall include at least the minimum number of occurrences indicated for each item in the appropriate table (i.e. Table 6. Card Application Data, Table 7. Administrative data items required at the Healthcard Server API and Table 8. Clinical data items required at the Healthcard Server API).
  - When emergency clinical information is read from the card, items containing each of the categories marked as mandatory in Tables 9 to 11 shall occur at least once. In each case, the structure shall conform with that specified for the appropriate item in Table 8 and the relevant indicator shall note the status of that item in relation to the card that has been read.

3.3.3 Card updating requirements

Systems that are able to update an interoperable healthcard data shall:

- Automatically update the Authorisation Details in the interoperable data when other interoperable data is updated.

3.3.4 Other requirements

The requirements stated above only cover aspects that are essential for interoperability. A healthcard issuer will also need to address other requirements such as:

- Software for issuing or updating cards
- Data structures for storage of information on the card. These may conform with ENV12018 or may be locally defined.
  - If data structures are developed to conform with ENV12018, additional attention should be paid to the proposals for representation of these logical data structures in IC smart cards. These proposals are not yet formally adopted but are the subject of active work in CEN TC224 WG12.
- Added value software which may provide additional facilities or enhance performance.

4 Healthcard Server API functions

4.1 Introduction

For the purposes of the interoperability demonstration the Healthcard Server API supports two functions.

HciReadData instructs the Healthcard Server to read a logical data set from the card.

HciGetData instructs the Healthcard Server to pass the logical data set read from the card to the API.

These functions are based on the CardLink functions Request_App_Data and Get_App_Data. Notes on the differences and the reasons for these differences are included in the specifications on the next pages.

The values returned by these functions are shown as global, typified constants and these are defined in Section 6. The generic data types used in function definitions in this specification are described in Section 7.3.

4.2 Open a Healthcard Server Manager session

UNIT16RC HciInitialise(*ResHdl, szHsConfig*)

HDL *ResHdl*; /* logical handle of an open CTM resource */

CHAR* *szHsConfig*; /* points to a zero terminated string that identifies a card configuration */

The **HciInitialise** loads and/or configures a selected Healthcard Server module. The loaded and configured Healthcard Server is associated with a particular resource indicated by *ResHdl*. The Healthcard Server does not need to be aware of the physical device to which *ResHdl* refers but must use this handle when communicating with the Card Terminal or card via the Card Terminal Manager API.

If the Healthcard Server supports several different healthcard formats, the Healthcard Server refers to configuration information indicated by *szHsConfig*. The method of configuration is Healthcard Server dependent. If the Healthcard Server is not configurable this parameter is ignored. The specified configuration must be used for processing subsequent Healthcard Server functions that refer to this resource handle (*ResHdl*), until another *HciInitialise* or an *HciTerminate* function request is received. The same Healthcard Server may be reinitialised with a different configuration but, before a Healthcard Server is loaded, any other Healthcard Server linked to the same resource handle must be terminated.

This function should be called whenever a new interoperable healthcard is recognised by the Healthcard Client Application (or an intermediate Healthcard Server Manager).

Parameter	I/O	Description
ResHdl	I	Resource handle allocated to a specified ICC slot by the Card Terminal Manager when the function CtmResOpen was called.
szHsConfig	I	An optional reference to Healthcard Server configuration information. This is recognised by the Healthcard Server and used to set configuration options. If the Healthcard Server is not configurable, this parameter is ignored.

Return codes

HCI_OK	
HCI_HANDLE_INVALID	The ResHdl is not a valid handle associated with an open CTM resource.
HCI_HANDLE_INUSE	The ResHdl is associated with an open link to another Healthcard Server. Use HciTerminate before proceeding.
HCI_SERVER_CONFIG_ERROR	The Healthcard Server did not accept the configuration associated with the recognised card.
HCI_CARD_ABSENT	The CTM found no card present during access.
HCI_CARD_MUTE	The card is not electrically connected and an attempt to power it up has failed.
HCI_CARD_OFF	The card is not electrically connected.
HCI_CARD_MISMATCH	The card cannot be read by this Healthcard Server.
HCI_SYSTEM_ERROR	An unspecified error has occurred in the Healthcard Server.
HCI_CTM_RESOURCE_LOCKED	The CTM tried to gain exclusive access but was denied or a CTM function was blocked by the exclusivity scope of another entity.
HCI_CTM_ERROR	An error has occurred in the CTM. The nature of this error

	may be reported using the error code

4.3 Close a Healthcard Server Manager session

UINT16RC HciTerminate(*ResHdl*)

HDL ResHdl; **/*** resource handle associated with an open link to this HS ***/**

The **HciTerminate** function closes a logical link between the Healthcard Client Application (or an intermediate Healthcard Server Manager) and the Healthcard Server.

If a Healthcard Server has no other open links it may be unloaded. This is dependent on the operating environment.

This function should be called when a period of interaction with a particular interoperable healthcard is completed.

Note that this function does not close the CTM resource, it only closes the association between this resource and the Healthcard Server. Thus if a particular Healthcard Server is found to be incapable of reading a card, another more suitable Healthcard Server can be loaded and configured.

Parameter	I/O	Description
ResHdl	I	The logical handle for the ICC slot associated with the open link with the Healthcard Server that is to be closed.

Return codes

HCI_OK	
HCI_HANDLE_INVALID	The ResHdl is not a valid handle associated with an open CTM resource.
HCI_HANDLE_INIT_ABSENT	The ResHdl is not associated with an open link to the Healthcard Server. Use HciInitialise first.
HCI_SYSTEM_ERROR	An unspecified error has occurred in the Healthcard Server.

4.4 Read Healthcard Interoperable Data

UINT16RC HciReadData(*ResHdl, nHciDataSet*)

HDL *ResHdl*; /* resource handle associated with an open link to the HSM */

UINT16 *nHciDataSet*; /* reference to an interoperable data set to be read */

The **HciReadData** function instructs the Healthcard Server to start reading a logical data set from the card in the slot referenced by the CTM resource handle *ResHdl*.

Parameter	**Description**
ResHdl	I The logical resource handle for the ICC slot through which the healthcard, that is to be read, is connected.
nHciDataSet	Refers to the interoperable data set to be read

Value	**Meaning**
0	Card Application Data
1	Administrative Data
2	Clinical Data

Return codes

HCI_OK	
HCI_HANDLE_INVALID	The ResHdl is not a valid handle associated with an open CTM resource.
HCI_HANDLE_INIT_ABSENT	The ResHdl is not associated with an open link to the Healthcard Server. Use HciInitialise first.
HCI_DATASET_INVALID	An invalid or unsupported data set has been specified in the calling parameters.
HCI_CARD_ABSENT	The CTM found no card present during access.
HCI_CARD_CHANGED	The card has been changed since the last call to CtmCardOpen referring to this ResHdl.
HCI_CARD_MUTE	The card is not electrically connected and an attempt to power it up has failed.
HCI_CARD_OFF	The card is not electrically connected.
HCI_SYSTEM_ERROR	An unspecified error has occurred in the Healthcard Server.
HCI_CTM_RESOURCE_LOCKED	The CTM tried to gain exclusive access but was denied or a CTM function was blocked by the exclusivity scope of another entity.
HCI_CTM_ERROR	An error has occurred in the CTM. The nature of this error may be reported using the error code

Comments

Before this function is called:

- The selected card must be inserted, reset and powered on using appropriate Card Terminal Manager API functions.
- The card must be recognised by the Healthcard Client Application (or an intermediate Healthcard Server Manager). The card operating system can be identified from the ATR (Answer To Reset). This is returned when the card is reset (see Card Terminal Manager API functions).
- A Healthcard Server that supports the card operating system of the selected card must be loaded and initialised using the HciInitialise function.
- The Healthcard Client Application (or an intermediate Healthcard Server Manager). can then read the Card Application Data using this function (nHciDataSet=0). The Card Application Data can then be used to select, load and/or configure an appropriate Healthcard Server.

This function does not itself return the data read from the card. The *HciGetData* function must be used to check the completion of reading, and to access the information that has been read. This means that the Healthcard Server API is asynchronous. Therefore, once the *HciReadData* command has been issued, the Healthcard Client Application can undertake other processing while awaiting the completion of the reading.

The *HciGetData* function can only retrieve the results of the last *HciReadData* function thus further *HciReadData* functions should not be submitted until the data has been retrieved.

Relationship to CardLink 1 specification

The CardLink 1 parameter “connex” is replaced by *ResHdl*. This is a handle obtained by the Healthcard Client Application when it opens a Card Terminal resource (such as an ICC slot) through the Card Terminal Manager API. This handle must be used by the Healthcard Server to access the appropriate Card Terminal and/or card through the Card Terminal Manager API.

4.5 Get Healthcard Interoperable Data

UINT16RC HciGetData(*ResHdl, nHciDataSet, nHciDataLen, pHciData*)

HDL *ResHdl*; /* resource handle associated with an open link to the HSM */
UINT16 *nHciDataSet*; /* reference to an interoperable data set to be read */
UINT32* *nHciDataLen*; /* points to the length of data set read or maximum buffer size */
CHAR** *pHciData*; /* points to a buffer containing data read from card */

The **HciGetData** function instructs the Healthcard Server to return the logical data set read from the specified card in a buffer. If reading is incomplete, the function returns HCI_READ_INCOMPLETE and no data is placed in the buffer.

Parameter	Description	
ResHdl	I	The logical resource handle for the ICC slot through which the healthcard, that is to be read, is connected.
nHciDataSet	Refers to the interoperable data set to be read	
	Value	**Meaning**
	0	Card Application Data
	1	Administrative Data
	2	Clinical Data
nHciDataLen	Input - maximum length of buffer at pHciData.	
	Output - actual length of data written to buffer at pHciData	
pHciData	Points to a buffer containing the data read from the card.	

Return codes

HCI_OK	
HCI_READ_INCOMPLETE	Reading is incomplete wait and retry.
HCI_HANDLE_INVALID	The ResHdl is not a valid handle associated with an open CTM resource.
HCI_HANDLE_INIT_ABSENT	The ResHdl is not associated with an open link to the Healthcard Server. Use HciInitialise first.
HCI_DATASET_INVALID	An invalid or unsupported data set has been specified in the calling parameters.
HCI_ DATASET_MISMATCH	The data set requested does not match the data set read from the card by the previous HciReadData.
HCI_BUFFER_FULL	The buffer allocated when the function was called is not large enough for the returned data. Call again with a larger buffer.
HCI_DATA_ERROR	The requested data could not be read from the card or when read from the card did not conform to the expected data structure and could not be mapped to the CTM-API structure.
HCI_CARD_ABSENT	The CTM found no card present during access.
HCI_CARD_CHANGED	The card has been changed since the last call to CtmCardOpen referring to this ResHdl.
HCI_CARD_MUTE	The card is not electrically connected and an attempt to

	power it up has failed.
HCI_CARD_OFF	The card is not electrically connected.
HCI_CARD_MISMATCH	The card cannot be read by this Healthcard Server.
HCI_SYSTEM_ERROR	An unspecified error has occurred in the Healthcard Server.
HCI_CTM_RESOURCE_LOCKED	The CTM tried to gain exclusive access but was denied or a CTM function was blocked by the exclusivity scope of another entity.
HCI_CTM_ERROR	An error has occurred in the CTM. The nature of this error may be reported using the error code

Comments

If this function is successful, it places the data read from the card by the last HciReadData function in a buffer indicated by pHciData and sets nHciDataLen to the total length of the data returned to the buffer. The data in the buffer is structured in accordance with the Tag-Length-Value format specified in Chapter 8.

The HciGetData function must be preceded by a HciReadData function referring to the same card and data set. If the last HciReadData function refers to a different data set, the value HCI_DATASET_MISMATCH is returned.

If the HciReadData function is still being processed the value HCI_READ_INCOMPLETE is returned. In this case, the Healthcard Client Application (or intermediate Healthcard Server Manager) should retry until the HCI_OK response is received. Thus the Healthcard Server API is asynchronous and once the HciReadData command has been issued the calling entity can undertake other processing while awaiting the completion of the reading.

The maximum buffer length is specified by nHciDataLen when the function is called. If the total length of the data exceeds this length the value HCI_BUFFER_FULL is returned. The HciGetData function must not write beyond the maximum size of the buffer and the Healthcard Server must return to its state prior to the HciGetData so that a subsequent use of the same function (with an adequate buffer) will return the appropriate information. The Healthcard Client Application (or intermediate Healthcard Server Manager) must **not** process the buffer if an error is returned as information may be absent, incomplete or inappropriate. To recover from this error the calling entity must specify a larger buffer and call the HciGetData function again.

The Healthcard Client Application (or intermediate Healthcard Server Manager) should not read the buffer indicated by pHciData unless the returned value is HCI_OK.

Relationship to CardLink 1 specification

The CardLink 1 parameter “connex” is replaced by *ResHdl*. This is a handle obtained by the Healthcard Client Application when it opens a Card Terminal resource (such as an ICC slot) through the Card Terminal Manager API. This handle must be used by the Healthcard Server to access the appropriate Card Terminal and/or card through the Card Terminal Manager API.

The parameter nHciDataLen has been added to allow the application to specify a limited sized buffer for the return on information.

The parameter nHciDataSet has been added to check synchronisation with the corresponding HciReadData command.

5 Card Terminal Manager API specification

5.1 Notes on previous versions

At the time of the EU Healthcard Interoperability Feasibility Study Consensus Meeting, in July 1996 two separate proposals were considered for the Card Terminal Manager API:-

a) the German MCT specification;

b) the proposals of the French GIP-CPS group.

The consensus meeting was informed that discussions between the two groups of experts were continuing, under the aegis of the TrustHealth project, with the intention of producing a common merged specification. It was agreed that if such a specification was available by the end of July 1996 it would be adopted for the EU Healthcard Interoperability demonstration between DiabCard and CardLink. It was also agreed that, unless the next meeting between GIP-CPS and MCT produced an outline specification of the required interface, an alternative, interim solution would need to be defined and implemented. Candidates for this were the CardLink 1 and MCT specifications.

As agreement was not reached within the envisaged timescale, an interim specification was included in the previous "final" draft of this specification. However, subsequently a good basis for agreement has been developed by MCT and GIP-CPS experts. As a result this revised specification has been produced including the same references to this specification as those included in the current G7 healthcard interoperability specification (version 0.5). The proposed Card Terminal Manager API specification fully supports the range of services that were identified in previous versions of the EU Healthcard Interoperability specification. It seems highly likely that it will form a more stable basis for development than the subset of the MCT included in previous versions. However, in view of the previously expressed views of those involved in technical development of the EU healthcard interoperability demonstrations, it is important to gain a broad consensus for this change rather than attempting to impose it.

Urgent comments on the way forward are particularly welcomed from those involved in developing demonstration products within the CardLink and DiabCard projects.

5.2 Introduction

The Card Terminal Manager API (CTM-API) is an interface that provides EU Healthcard Servers and CAM Middleware with access to cards and card readers. Ideally the interface should fully support the functions currently specified for both the CAM Reader Device Driver API (as this is the interface expected by CAM Middleware) and by the EU Card Terminal API (as this is the interface expected by EU Healthcard Servers).

Eventually a common Card Terminal Manager API should be agreed as an International Standard. However, this is unlikely to happen within the timescale of current EU and G7 healthcard interoperability activities. At present, there are several proposals for "common" interfaces to different Card Terminals. These include the German Multifunction Card Terminal (MCT), the French GIP-CPS "Gestionnaire", the CardLink card reader interface and the Japanese CAM Card Reader Device Driver (CAM-CRDD). The Card Terminal Manager functions proposed in this healthcard interoperability specification are a subset of the proposals that have arisen from discussions between GIP-CPS and MCT experts. The subset documented here provides the essential services identified in earlier drafts of the specification.

The full list of functions proposed for the joint work of GIP-CPS and MCT experts are listed below. The functions shown in bold are the minimum that need to be supported for the purposes of healthcard interoperability. The two functions shown in italics represent proposed lower-level functions used to exchange data and provide more flexible versions of the basic functions. The exclusivity setting function (shown in bold italics) is likely to be mandatory in France and Germany. Its use is strongly recommended in interoperable healthcard systems.

- **CtmOpen : Open session with CT-Manager;**
- **CtmClose : Close session with CT-Manager;**
- **CtmResOpen : Open logical link with a Card Terminal resource;**
- CtmResClose : Close logical link with a Card Terminal resource;
- ***CtmSetExlusivity : Set exclusivity;***
- CtmCtReset : Card Terminal Reset;
- CtmCtAutoTest : Test the Card Terminal;
- CtmCtApplicationOpen : Open logical link with an application in a CT;
- CtmCtApplicationClose : Close logical link with an application in a CT;
- CtmCtApplicationExchange : Message exchange with an application in a CT;
- *CtmCtExchange : Card Terminal exchange data;*
- **CtmCardOpen : Request Card;**
- **CtmCardClose : Power Off and Remove Card;**
- **CtmCardTest : Test Card;**
- **CtmCardCommand : Transparent command exchange with card;**
- *CtmResExchange : Resource Exchange.*

This remainder of this section deals only with the proposed minimum set of CTM-API services required for initial healthcard interoperability. The full draft of the current proposals is available as an annex to this document.

The generic data types used in function definitions in this specification are described in Section 7.3.

5.3 Resource Session Management

5.3.1 CtmOpen : Open session with CT-Manager

UINT16RC CtmOpen(pAppHdl)

HDL* *pAppHdl*; /* points to logical handle AppHdl used by CTM to identify calling entity */

The **CtmOpen** function establishes a logical link between a calling entity and the Card Terminal Manager. A handle is returned by the Card Terminal Manager. This handle must be used by the calling entity when it opens resources and when closing its link to the Card Terminal Manager.

This handle allows the CT-Manager to identify the calling entity when it opens resources in order to keep track of its logical links.

Parameter	I/O	Description
AppHdl	O	Returned handle that uniquely identifies the calling entity at the CT-Manager level

Return codes

CTM_OK	
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager

Comments

An application must only call this function once. Usually it is called when an application starts up. If middleware that works as part of an application needs to separately open Card Terminal resources, it should be informed of the AppHdl by the calling application and should not call CtmOpen itself.

Warning : if an application (or middleware being part of this application) calls CtmOpen twice, it will receive 2 different handles. This is to be avoided because there is a potential conflict when exclusivity is requested on 2 resources each opened with a different AppHdl.

Usage in healthcard interoperability

The CtmOpen function is called once by the Healthcard Client Application when it is loaded.

5.3.2 Close session with CT-Manager

UINT16RC CtmClose(AppHdl)

HDL *AppHdl*; /* logical handle identifying the calling entity to the CTM */

The **CtmClose** function closes a logical link between the calling entity and the Card Terminal Manager. The AppHdl specified in the function call is released within the Card Terminal Manager. Any resources opened using the specified AppHdl are closed and any associated exclusivities are released. The state of any card inserted in a slot referred to by a resource that is closed using this function is not altered (i.e. if the card is powered-up it remains powered-up).

Parameter	I/O	Description
AppHdl	I	Handle returned when the calling entity opened a logical link with the CTM-API using the CtmOpen function.

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager

Comments

An application that has called the CtmOpen function must call this before terminating to avoid leaving redundant handles and exclusivity tables in the Card Terminal Manager.

Usage in healthcard interoperability

The CtmOpen function is called once by the Healthcard Client Application when it exits.

5.3.3 Open logical link with a Card Terminal resource

UINT16RC CtmResOpen(AppHdl, szResName, pResHdl)

HDL *AppHdl*; /* logical handle identifying the calling entity to the CTM */

CHAR* *szResName*; /* zero terminated string logical name of the resource */

HDL* *pResHdl*; /* points to ResHdl, a returned handle identifying the resource */

The **CtmResOpen** function establishes a logical link between the calling entity and the referenced resource. A referenced resource is usually an ICC slot. However, in some cases it may be a display or keyboard associated with an ICC slot.

The ResHdl returned to the calling entity is used for all subsequent accesses to that resource or to the Card Terminal in which that resource is installed.

Note : no I/O is performed with the referenced resource.

Parameter	I/O	Description
AppHdl	I	Handle returned when the calling entity opened a logical link with the CTM-API using the CtmOpen function.
szResName	I	Predefined logical name (string containing ASCII characters and terminated by a binary zero) of a resource referencing configuration information. The name may be allocated by an application provided that an installation program links the name to the appropriate physical resource by updating configuration information.
ResHdl	O	Returned handle that uniquely identifies the resource for use in subsequent functions at the CTM-API level.

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager
CTM_RESOURCE_ALREADY_OPEN	The calling application has already opened the specified resource.
CTM_NO_CONFIG_INFO	The specified configuration name does not match any of the Card Terminal resource configurations on the system.
CTM_UNKNOWN_RESOURCE	The type of resource specified is not recognised.
CTM_PARAMETER_INVALID	A parameter to a CTM-API function was invalid

Comments

An application should call this function once for each logical resource that it may need to access. It should usually do this when an application starts up. CtmOpen must be called before its first call to CtmResOpen.

Middleware that works as part of an application should be informed of the ResHdl of any resources that it is sharing with its calling application.

Middleware should only call CtmResOpen itself (after a CtmOpen) if it needs to establish a level of exclusivity while preventing its calling application from accessing to that same resource.

Usage in healthcard interoperability

The CtmResOpen function is called once by the Healthcard Client Application when it loads to open the ICC slot in which interoperable healthcards will be inserted.

5.3.4 Set exclusivity

UINT16RC CtmSetExclusivity(ResHdl, ExclScope)

HDL *ResHdl*; /* logical handle identifying the resource */

UINT8 ExclScope; /* an indication of the extent of exclusivity required */

The **CtmSetExclusivity** function establishes exclusive control for the calling entity over the referenced resource to the extent specified by the exclusivity scope.

The function may be used to gain exclusive control of:

- A specified resource (e.g. an ICC slot), or;
- The Card Terminal in which that resource is located, or;
- The physical port through which that Card Terminal is connected to the host.

The function may also be used to clear the exclusivity scope.

When a resource is opened using CtmResOpen, the exclusivity scope for the calling entity is set as zero.

When a request to set exclusivity is received, the Card Terminal Manager checks that the specified resource is not already within the scope of an exclusivity setting established by another calling entity.

Note : Functions already accepted before the request for exclusivity will be entirely executed.

When exclusivity is accepted, the appropriate setting is made in a table maintained by the Card Terminal Manager.

Parameter	I/O	Description
ResHdl	I	Handle returned when the calling entity opened a logical link with the resource using the CtmResOpen function.
ExclScope	I	0 - No exclusivity; 1 - Exclusive control over the resource 2 - Exclusive control over the resource and the Card Terminal in which it is located 3 - Exclusive control over the resource, the Card Terminal in which it is located and the port through which this Card Terminal is connected to the host.

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager
CTM_EXCLUSIVITY_REFUSED	A request for exclusive access could not be met.
CTM_PARAMETER_INVALID	A parameter to a CTM-API function was invalid

Comments

Exclusivity is used in a defensive manner to protect an application from interference with its access to a Card Terminal resource. Provided another calling entity does not have exclusive control over a resource any CTM-API function can be called without first setting exclusivity.

Exclusivity is only required if a sequence of commands has to be executed while maintaining context. Exclusivity should be set to the lowest level compatible with maintaining the required context and should be reset to zero as soon as the sequence of commands that required protection is completed.

Note 1 : An application working alone on a workstation and which does not use exclusivity protection may fail or treat incorrect data (without knowing it) when another card application is installed on that workstation.

Note 2 : If this function is to be called more than once, and if one CtmSetExclusivity is refused, all previous accepted exclusivities shall be released before retrying the whole sequence. Therefore it is good practice to group the CtmSetExclusivity calls.

Usage in healthcard interoperability

The CtmSetExclusivity function is called by a Healthcard Server (or by other Healthcard Middleware) before and after a sequence of card commands. Before the sequence, the scope is set to one to give exclusive control over the resource, and after each sequence the scope is reset to zero to release exclusivity.

5.4 Resource functions

5.4.1 General comment

The Resource functions in this section will be blocked by the Card Terminal Manager if any other calling entity has an exclusivity scope that includes that resource.

5.4.2 Request or check card insertion

UINT16RC CtmCardOpen (ResHdl, SzPrompt, EffectFlags, TimeOut, plenATR, pATR, pCardState, pStatus)

```
HDL ResHdl;          /* logical handle identifying the resource* /
CHAR* SzPrompt;      /* zero terminated prompt string or null pointer* /
UINT32 EffectFlags;  /* set of binary flags controlling CT effects */
UINT8 TimeOut;  /* time out in seconds */
UINT16* plenATR;     /* points to length of returned ATR or maximum buffer length */
CHAR** pATR;         /* points to the ATR return buffer */
UINT8* pCardState;   /* points to CardState – an indication of the state of the card */
UINT16* pStatus;     /* points to Status – an Indication of CT execution status */
```

The **CtmCardOpen** function first checks the ICC slot associated with the referenced resource.

If a card is present and already powered on – the appropriate CardState is returned together with the ATR established when the card was initially reset. A new reset is NOT performed.

If a card is present but not powered on – the card is powered on and a reset is performed. The appropriate CardState and ATR are then returned. In this case the Card-Terminal Manager sets the "card-changed" marker for all other resources that refer to the same physical device. The CardState is reported as "card-changed" but then the "card-changed" marker for this resource handle is cleared.

If no card is present – prompts and/or effects (as specified in the function call) are used to request the user to insert a card. When a card is inserted it is powered up and the CardState and ATR are returned to the calling entity. When a card is inserted the Card Terminal Manager sets the "card-changed" marker for all resources that refer to this physical device.

If no card is inserted within the specified period of time – the appropriate status and CardState are reported and CTM_STATUS_ERROR is returned. In this case, the Card-Terminal Manager sets the "card-changed" marker for all resources that refer to this physical device.

Parameter	I/O	Description
ResHdl	I	Handle returned when the calling entity opened a logical link with the resource using the CtmResOpen function.
SzPrompt	I	A text string or null pointer. - Null pointer = No prompt, (default value if tag absent) - Blank string = Default prompt for insertion of card, - Non blank = String displayed as prompt. Prompts are ignored if the Card Terminal does not support them. (e.g. if the CT does not include a display). No error is reported in these cases.

EffectFlags	I	Integer representation of binary flags: Bit 1 (lsb) ignored, Bit 2 set (value 2) = acoustic prompt (e.g. beep), Bit 3 set (value 4) = optical prompt (e.g. LED). Bit 4 to Bit 8 (msb) ignored Thus if value of EffectFlags is 6 then both acoustic and optical prompts are used. Effect flags are ignored if the Card Terminal does not support the effect. No error is reported in these cases.
TimeOut	I	0 : No wait for card insertion (if present immediate power-up (default value if tag absent) 1 to 254 : Time in seconds to wait for card insertion 255 : Wait for card insertion, no limit
lenATR	I	Maximum length of ATR buffer
	O	Actual length of ATR data returned.
ATR	O	The full ATR returned by the card.
CardState	O	No card 0 (all bits clear) CTM_EXEC_TIMEOUT Card inserted power n/k 1 (b1) ? CTM_EXEC_OK* Card inserted no power 3 (+b2) CTM_EXEC_CARD_MUTE Card powered (unchanged) 5 (+b3) CTM_EXEC_OK* Card powered (changed) 133 (+b8 set) CTM_EXEC_OK* *or CTM_EXEC_CARD_UNKNOWN
Status	O	CTM_EXEC_OK CTM_EXEC_PARAMETER_INVALID CTM_EXEC_REFUSED_BY_CT CTM_EXEC_COMM_ERROR CTM_EXEC_ABORTED (process aborted by cancel-key on CT) CTM_EXEC_CARD_ABSENT CTM_EXEC_CARD_UNKNOWN (protocol not supported) CTM_EXEC_CARD_MUTE CTM_EXEC_TIMEOUT (No card presented within specified time)

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager
CTM_PARAMETER_INVALID	A parameter to a CTM-API function was invalid
CTM_BLOCKED_EXCLUS_SETTING	An exclusivity setting held by another calling entity prevents completion of the requested function.
CTM_STATUS_ERROR	status parameter is not CTM_EXEC_OK

Usage in healthcard interoperability

Used by Healthcard Client Application or Healthcard Server Manager to reset a newly inserted card.

5.4.3 Power-off and enable card removal

UINT16RC CtmCardClose(ResHdl, SzPrompt, EffectFlags, TimeOut, pCardState, pStatus)

```
HDL ResHdl;            /* logical handle identifying the resource* /
CHAR* SzPrompt;        /* zero terminated prompt string or null pointer* /
UINT32 EffectFlags;    /* set of binary flags controlling CT effects */
UINT8 TimeOut;  /* time out in seconds */
UINT16* plenATR;       /* points to length of returned ATR or maximum buffer length */
CHAR** pATR;           /* points to the ATR return buffer */
UINT8* pCardState;     /* points to CardState – an indication of the state of the card */
UINT16* pStatus;       /* points to Status – an Indication of CT execution status */
```

The **CtmCardClose** function powers off the card in the ICC slot associated with the referenced resource, and optionally prompt for removal of the card by the user or eject the card.

The CtmCardClose function first checks the ICC slot associated with the referenced resource.

If no card is present – the CardState is returned.

If a card is present – the card is powered off, if it was powered-on.

If no removal/ejection is requested the appropriate CardState is returned.

If prompts and/or effects are used (as specified in the function call) to eject the card or request the user to remove the card, the appropriate CardState is returned when the card is removed.

If prompts and/or effects are used (as specified in the function call) to eject the card or request the user to remove the card and if a card is still present in the slot after the specified period of time – the appropriate status and CardState are reported and CTM_STATUS_ERROR is returned.

Unless the function is blocked, the Card-Terminal Manager forces the CardState states of all sessions with cards in the addressed Card-Terminal to "card-changed".

Parameter	I/O	Description
ResHdl	I	Handle returned when the calling entity opened a logical link with the resource using the CtmResOpen function.
SzPrompt	I	A text string or null pointer. - Null pointer = No prompt, (default value if tag absent) - Blank string = Default prompt for insertion of card, - Non blank = String displayed as prompt. Prompts are ignored if the Card Terminal can not treat them (Ex : if the CT does not include a display). No error is reported in these cases.
EffectFlags	I	Integer representation of binary flags: Bit 1 set (value 1) = mechanically eject the card, Bit 2 set (value 2) = acoustic prompt (e.g. beep), Bit 3 set (value 4) = optical prompt (e.g. LED). Bit 4 to Bit 8 (msb) ignored Thus if value of EffectFlags is 6 then both acoustic and optical prompts are

		used. Effect flags are ignored if the Card Terminal does not support the effect. No error is reported in these cases.		
TimeOut	I	0 : No wait for card insertion (if present immediate power-up (default value if tag absent) 1 to 254 : Time in seconds to wait for card insertion 255 : Wait for card insertion, no limit		
CardState	O	No card	0 (all bits clear)	CTM_EXEC_OK
		Card inserted power n/k	1 (b1)	CTM_EXEC_TIMEOUT
		Card inserted no power	3 (+b2)	CTM_EXEC_TIMEOUT
		Card powered (unchanged)	5 (+b3)	CTM_EXEC_REFUSED_BY_CT
		Card powered (changed)	133 (+b8 set)	CTM_EXEC_REFUSED_BY_CT
Status	O	CTM_EXEC_OK CTM_EXEC_PARAMETER_INVALID CTM_EXEC_REFUSED_BY_CT CTM_EXEC_COMM_ERROR CTM_EXEC_TIMEOUT (No card presented within specified time)		

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager
CTM_PARAMETER_INVALID	A parameter to a CTM-API function was invalid
CTM_BLOCKED_EXCLUS_SETTING	An exclusivity setting held by another calling entity prevents completion of the requested function.
CTM_STATUS_ERROR	status parameter is not CTM_EXEC_OK

Usage in healthcard interoperability

Used by Healthcard Client Application to enable removal of an interoperable healthcard.

5.4.4 Test the state of a card or ICC slot

UINT16RC CtmCardTest(ResHdl, pCardState, pStatus)

HDL *ResHdl*; /* logical handle identifying the resource* /

UINT8* *pCardState;* /* points to CardState – an indication of the state of the card */

UINT16* *pStatus;* /* points to Status – an Indication of CT execution status */

The **CtmCardTest** function checks the ICC slot associated with the referenced resource and returns the appropriate CardState.

Parameter	I/O	Description		
ResHdl	I	Handle returned when the calling entity opened a logical link with the resource using the CtmResOpen function.		
CardState	O	No card	0 (all bits clear)	CTM_EXEC_OK
		Card inserted power n/k	1 (b1)	CTM_EXEC_OK
		Card inserted no power	3 (+b2)	CTM_EXEC_OK
		Card powered (unchanged)	5 (+b3)	CTM_EXEC_OK
		Card powered (changed)	133 (+b8 set)	CTM_EXEC_OK
Status	O	CTM_EXEC_OK CTM_EXEC_PARAMETER_INVALID CTM_EXEC_REFUSED_BY_CT CTM_EXEC_COMM_ERROR		

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager
CTM_PARAMETER_INVALID	A parameter to a CTM-API function was invalid
CTM_BLOCKED_EXCLUS_SETTING	An exclusivity setting held by another calling entity prevents completion of the requested function.
CTM_STATUS_ERROR	status parameter is not CTM_EXEC_OK

Usage in healthcard interoperability

Used by a Healthcard Client Application or by a G7 Healthcard Server Manager to check that a card has not been changed prior to reading an interoperable data set from the card.

5.4.5 Transparent command exchange with card

UINT16RC CtmCardCommand(ResHdl, lenCommand, pCommand, plenResponse, pResponse, pCardState, pStatus)

HDL *ResHdl*;	/* logical handle identifying the resource* /
UNIT32 *lenCommand*	/* length of command * /
CHAR* *pCommand*	/* points to card Command * /
UNIT32* *lenResponse*	/* points to length of response or maximum length of response buffer* /
CHAR** *pResponse*	/* point to Response buffer * /
UINT8* *pCardState;*	/* points to CardState – an indication of the state of the card */
UINT16* *pStatus;*	/* points to Status – an Indication of CT execution status */

The **CtmCardCommand** function exchanges a command with the card inserted in the ICC slot associated with the referenced resource.

If card-changed since last CtmRequestCard for this ResHdl, the command is not sent to the card. In this case the Status and CardState report the appropriate values and the return code is CTM_STATUS_ERROR. If the application wishes to continue to process the card after this error, it must first call the CtmRequestCard function to set a new context for the exchange.

If at return the CardState indicates "card-changed", the Card-Terminal Manager forces the CardState of all sessions with cards in the addressed Card-Terminal to "card-changed".

Parameter	I/O	Description		
ResHdl	I	Handle returned when the calling entity opened a logical link with the resource using the CtmResOpen function.		
lenCommand	I	Length of the Command		
Command	I	The ICC Command as an octet string conforming to ISO7816.		
lenResponse	I	Maximum length of Response buffer		
	O	Actual length of Response.		
Response	O	Response buffer. The response as an octet string conforming to ISO7816 including SW1 and SW2.		
CardState	O	No card	0 (all bits clear)	CTM_EXEC_CARD_ABSENT
		Card inserted power n/k	1 (b1)	CTM_EXEC_OK
		Card inserted no power	3 (+b2)	CTM_EXEC_CARD_MUTE
		Card powered (unchanged)	5 (+b3)	CTM_EXEC_OK
		Card powered (changed)	133 (+b8 set)	CTM_EXEC_CARD_CHANGED
Status	O	CTM_EXEC_OK CTM_EXEC_PARAMETER_INVALID CTM_EXEC_REFUSED_BY_CT CTM_EXEC_COMM_ERROR CTM_EXEC_CARD_CHANGED CTM_EXEC_CARD_ABSENT CTM_EXEC_CARD_UNKNOWN (protocol not supported)		

CTM_EXEC_CARD_MUTE

Return codes

CTM_OK	
CTM_HANDLE_INVALID	The handle specified does not relate to an open CTM resource.
CTM_ERROR	An unspecified error has occurred in the Card Terminal Manager
CTM_PARAMETER_INVALID	A parameter to a CTM-API function was invalid
CTM_BLOCKED_EXCLUS_SETTING	An exclusivity setting held by another calling entity prevents completion of the requested function.
CTM_STATUS_ERROR	status parameter is not CTM_EXEC_OK

Usage in healthcard interoperability

Used by Healthcard Server (or other Healthcard Middleware) to read interoperable data from the card.

Also used by a G7 Healthcard Server Manager to confirm the identity of a CAM card.

6 The Card Interface

6.1 Communication protocol

The Card Interface for the interoperability pilot will support the following cards that conform with ISO 7816 parts 1 to 4:

- Any cards that fully implement the T=0 protocol;
- Cards that support the T=1 protocol subject to limitations imposed by different interpretations of some optional parameters in T=1 as defined by ISO7816. In these cases initial demonstration may be limited to cards that implement T=1 in a manner that conforms with an interpretation of ISO 7816 by one or more industrial partners in EU funded healthcard projects.[2]

6.2 Answer to Reset (ATR)

The ATR returned by the card must conform to ISO 7816 part 3.

The pre-issuing Data Object in the historical bytes of the ATR is used to identify the manufacturer and card operating system.

The ATR may be unable to identify the issuing application or healthcard format because:

- Several issuers and healthcard formats may use cards from the same manufacturer and with the same mask. In most cases, the mask determines the ATR. While large scale implementations may specify a healthcard specific mask smaller projects and implementations will not justify this process.
- In future multifunction cards may contain several applications. The ATR is unable to present multiple instances of application information in an unambiguous manner.

Therefore the ATR is used to determine the card operating system and thereafter the Card Application data is read from the card. It follows that for any card operating system (as identified by the ATR) there must be a common method of obtaining the Card Application data. This common method must be:

- Used on interoperable healthcards that use the same card operating system
- Documented and available to all developers who may implement their healthcard application on these cards.
- Implemented by all Healthcard Servers that support that card operating system.

6.3 Relationship to German MCT

The full MCT specification supports asynchronous protocols that conform with ISO 7816 and synchronous IC (memory) card protocols applicable to the German Health Insurance card. While it is considered desirable to support this wider range of card technologies, it is not essential for initial healthcard interoperability demonstrations.

---

[2] Consultations suggest that there are differences in the interpretation of the T=1 protocol by different suppliers. However, the interoperability should be based wherever possible on a common subset that is supported by the widest range of suppliers. It is anticipated that further refinement of ISO7816 will improve interoperability between different cards.

7 Additional reference information

7.1 Global, typified constants

The following global, typified constants are used to specify the values returned by the functions described in this specification.

7.1.1 EU Healthcard Server Return Codes

Code	Typified constant	Meaning
0	**HCI_OK**	
1	**HCI_CARD_ABSENT**	The CTM found no card present during access.
3	**HCI_CARD_CHANGED**	The card has been changed since the last call to CtmCardOpen referring to this ResHdl.
4	**HCI_TIMEOUT**	A function could not be completed within a predefined timeout.
5	**HCI_HANDLE_INVALID**	The ResHdl is not a valid handle associated with an open CTM resource.
6	**HCI_HANDLE_INUSE**	The ResHdl is associated with an open link to another Healthcard Server. Use HciTerminate before proceeding.
7	**HCI_CARD_MISMATCH**	The card cannot be read by this Healthcard Server.
9	**HCI_CARD_MUTE**	The card is not electrically connected and an attempt to power it up has failed.
10	**HCI_CARD_OFF**	The card is not electrically connected.
11	**HCI_HANDLE_INIT_ABSENT**	The ResHdl is not associated with an open link to the Healthcard Server. Use HciInitialise first.
12	**HCI_SERVER_CONFIG_ERROR**	The Healthcard Server did not accept the configuration associated with the recognised card.
13	**HCI_SYSTEM_ERROR**	An unspecified error has occurred in the Healthcard Server.
14	**HCI_DATASET_MISMATCH**	The data set requested does not match the data set read from the card by the previous HciReadData.
15	**HCI_READ_INCOMPLETE**	Reading is incomplete wait and retry.
16	**HCI_BUFFER_FULL**	The buffer allocated when the function was called is not large enough for the returned data. Call again with a larger buffer.
17	**HCI_DATASET_INVALID**	An invalid or unsupported data set has been specified in the calling parameters.
18	**HCI_DATA_ERROR**	The requested data could not be read from the card or when read from the card did not conform to the expected data structure and could not be mapped to the CTM-API structure.
31	**HCI_CTM_RESOURCE_LOCKED**	The CTM tried to gain exclusive access but was denied or a CTM function was blocked by the exclusivity scope of another entity.
32+	**HCI_CTM_ERROR**	An error has occurred in the CTM. The nature of this error may be reported using the CTM error code and adding 32.

7.1.2 Card Terminal Manager Return Codes

Code	Typified constant	Meaning
0	**CTM_OK**	
1	**CTM_HANDLE_INVALID**	The handle specified does not relate to an open CTM resource.
2	**CTM_RESOURCE_ALREADY_OPEN**	The calling application has already opened the specified resource.
3	**CTM_NO_CONFIG_INFO**	The specified configuration name does not match the Card Terminal resource configurations on the system.
4	**CTM_UNKNOWN_RESOURCE**	The type of resource specified is not recognised.
5	**CTM_PARAMETER_INVALID**	A parameter to a CTM-API function was invalid
6	**CTM_UNKNOWN_FUNCTION**	The function specified is not recognised.
7	**CTM_EXCLUSIVITY_REFUSED**	A request for exclusive access could not be met.
8	**CTM_BLOCKED_EXCLUS_SETTING**	An exclusivity setting held by another calling entity prevents completion of the requested function.
9	**CTM_ERROR**	An unspecified error has occurred in the CTM
254	**CTM_STATUS_ERROR**	The status code indicates a status that prevent successful completion of the function.

7.1.3 Card Terminal Manager Status Codes

Code	Typified constant	Meaning
0	**CTM_EXEC_OK**	
4	**CTM_EXEC_PARAMETER_INVALID**	The command sent to the Card Terminal included invalid parameters.
5	**CTM_EXEC_REFUSED_BY_CT**	The Card Terminal has refused the command for reasons not known by the Card Terminal Manager.
6	**CTM_EXEC_COMM_ERROR**	A communication error occurred during dialog with the Card Terminal
9	**CTM_EXEC_ABORTED**	The user aborted the function at the Card Terminal.
10	**CTM_EXEC_CARD_ABSENT**	No card in the specified slot.
11	**CTM_EXEC_CARD_UNKNOWN**	The card uses a communication protocol that is not supported by the Card Terminal.
12	**CTM_EXEC_CARD_MUTE**	The card is not electrically connected.
13	**CTM_EXEC_TIMEOUT**	A timeout occurred while waiting for completion of a Card Terminal function.

7.2 Configuration files

Two configuration files are defined in this specification. One of these is used when recognising cards and selecting the appropriate Healthcard Server, the other is used by the Card Terminal Manager.

7.2.1 Healthcard Server Manager Configuration file

The Healthcard Server Manager Configuration file (GHI-HSM.INI) includes details of the Healthcard Servers available. For each Healthcard Server, data is provided that details the name of the library file and the factors by which appropriate healthcards are recognised (e.g. ATR and Card Application Identifiers).

This file is read by the Healthcard Client Application (or an intermediate Healthcard Server Manager) to provide a template for matching an inserted card against the available Healthcard Servers. If a match is found, the appropriate Healthcard Server is then loaded to allow the card to be read.

Suppliers of healthcards and Healthcard Servers must provide the relevant entries in a form that can be appended to an existing GHI-HSM.INI file.

Table 1. EU Healthcard Server Configuration Data

[<Configuration Group Name>]	An arbitrary name used to group together the configuration data relating to a particular type of healthcard supported by the Healthcard Server.
Issuer=<*CountryCode*><*IssuerIdentifier*>	The country code and issuer code of healthcards supported by this configuration.
ATR=<*Historical bytes of ATR in Hex*>	The ATR historical bytes returned by cards of this type.
CardOS=<*Name of card OS*>	OPTIONAL: The name and version of the card operating system corresponding to the ATR entry.
CardApplicationIdentifier = <*apid*>	<apid> should match the CardApplicationIdentifier of cards supported by this configuration. This is the Application Identifier (AID) as defined by ISO 7816-5. It is presented in the configuration file and at the Healthcard Server API as a string of hexadecimal characters. The length of this string varies in accordance with the registration categories defined in ISO 7816-5 but will not exceed 32 characters. There may be multiple instances of this item if the same configuration is appropriate for several variations of the same application.
Version=<*CardApplicationVersion*>	OPTIONAL: The CardApplicationVersion number(s) in the CardApplicationData of cards supported by this configuration. If several versions are supported these may be expressed as a range with word "TO" between the first and last supported versions.
Library = <*libfile*>	<libfile> is the name of the program or program library that includes the HciReadData and HciGetData functions for cards supported by this configuration.
Configuration=<*HsConfig*>	OPTIONAL: <HsConfig> is used as the szHsConfig parameter when calling the HciInitialise function to configure an EU Healthcard Server. This is used if a single Healthcard Server can be configured to work with different healthcards. The format of the configuration file is dependent on the Healthcard Server software.

7.2.2 Card Terminal Manager Configuration file

Card Terminal Managers require configuration information including details of the local set up of Card Terminals and the appropriate driver software.

This information associates a logical name with:

- A Card Terminal;
- The host software driver that should be loaded to address that Card Terminal;
- Other information required by driver software on the host to allow it to address a particular slot in a particular Card Terminal. This information will vary according to the capabilities of the Card Terminal and the specification of the driver software. It may include:
  - The physical port or address through which the host is linked to that Card Terminal;
  - The sub-address if more than one Card Terminal is connected to a single port;
  - The functional unit number of a particular slot in that Card Terminal;
  - The functional unit number of the Card Terminal itself for receiving basic commands;
  - The functional unit number of any attached screen or keyboard.

This information is checked by the Card Terminal Manager when it opens a link to a particular slot in a Card Terminal. The relevant information about the specified slot is used to allow the Card Terminal Manager to pass the correct parameters to the CT-API in response to subsequent CTM-API functions.

The Card Terminal Manager configuration file must be updated:

- When installing a new Card Terminal or changing the connection between the host and the Card Terminal.
- When upgrading Card Terminal driver software.
- When installing an application that uses a particular logical name to refer to a resource.

Card Terminal Manager configuration data must be held in a form that is accessible to the Card Terminal Manager. As an illustration the Card Terminal Manager configuration information could be stored in file called "GHI-CTM.INI". However, the decision on actual storage is left to the designers of Card Terminal Managers.

The file may contain various information related to the Card Terminals attached to the system. As a minimum it should contain the following entries:

Table 2. Specification of the Card Terminal Configuration File

[<szResName>]	Matches the szResName parameter in the CtmResOpen function.
Port = <pn>	Port number to which the Card Terminal is connected
CtAddress = <cta>	Address of Card Terminal in a chain of devices on the same port
CtUnit = <fu>	Functional unit (e.g. slot number) within the Card Terminal
CtDriver = <dd>	Lower-level device driver required to provide CT-API for access to this Card Terminal.

Table 3. Example Card Terminal Configuration File

[G7_PATIENT_CARD]	; Following configuration is used for G7 patient cards.
Port = COM1	; Card Terminal 1 is connected to COM1

```
CtAddress = 0          ; First or only connection on a chain
CtUnit = 1             ; this ICC slot is functional unit 1
CtDriver = MCT09.DLL   ; Use a driver called MCT09.DLL to access the Card Terminal.
```

Note: The parameters required for Card Terminal services may vary according to the requirements of the Card Terminal and its driver software. In some case, certain parameters may be unnecessary. For example, if a single slot Card Terminal is used with no "chaining", the CtAddress and CtUnit may be set by default rather than specified in a configuration file.

7.3 Generic types

To keep consistent with different development environments, some generic data types are used in this document. They will have to be further defined for each implementation :

UINT8	8 bit coded unsigned integer.
UINT16	16 bit coded unsigned integer.
UINT32	32 bit coded unsigned integer.
UINT8*	Pointer to an 8 bit coded unsigned integer.
UINT16*	Pointer to a 16 bit coded unsigned integer.
UINT32*	Pointer to a 32 bit coded unsigned integer.
UNIT16RC	A return code represented as a 16 bit encoded integer.
HDL	Value returned by the CT Manager to identify an entity. Unless otherwise specified in an implementation this is implemented as a UINT16
CHAR*	A pointer to a zero terminated string of octet characters.
CHAR**	A pointer to a pointer that is set by a function to point to a buffer of octet characters. The length of the buffer is specified by another parameter.

8 Data content and structures

8.1 Introduction

8.1.1 Existing data sets

There have been many attempts to agree a standard set of data. The two most authoritative are those in ENV12018 and the EUROCARDS WG4 report. In preparing this chapter, account has been taken of these documents and of the existing CardLink and DiabCard data sets. The CardLink set is based on the EUROCARDS work but includes some modifications required to meet specific needs. The DiabCard data set focuses on the specific needs of diabetes but also contains administrative and emergency clinical data.

Several attempts have been made to contrast different administrative and emergency data sets and derive a common set. Information provided by Jürgen Sembritzki and Claudia Hildebrand has been particularly helpful as a source for these proposals. The data sets vary in several ways including:

- Content - i.e. data items may or may not be recordable;
- Detail - i.e. inclusion of date, author and other data as well as the mere presence of an emergency condition;
- Specificity - i.e. the form of representation may allow an item to be recorded in general or specific terms (e.g. “Beta-blocker drugs” or “Propranolol 40mg tabs three daily”).

If all the variations in individual data sets are taken into account, the true common set is too small to be useful. Therefore, it is necessary to take a proactive approach to agree an interoperable data set which extends beyond the overlaps of existing data sets.

8.1.2 Enabling and mandating data sets

Data set determination depends on the clinical issue of what information is necessary or desirable and the technical issue of how this information is stored and presented. There has already been considerable clinical input into the current data sets used by CardLink and DiabCard. The role of this feasibility study is not to replace or undermine the evolving clinical consensus but to specify a technical solution that enables interoperability while accommodating current requirements and allowing extension to meet evolving requirements.

Therefore rather than proposing a definitive mandatory data set, we recommend an approach that defines the manner in which interoperable data should be presented at the API. We also propose a common subset of interoperable data that should be used to demonstrate the effectiveness of this approach in the CardLink 2 and DiabCard 3 interoperability pilots.

The data set proposed comprises some mandatory elements and some optional elements. In the case of an optional element that is not supported by a particular healthcard, the Healthcard Server should inform the Healthcard Client Application that the specified information is unsupported.

The inclusion of optional elements is unnecessary for the initial demonstration of interoperability between CardLink and DiabCard. We could instead limit the data set to items supported by both cards and declare all these items as mandatory. However, this would obstruct progress, as any extension of the data set would render existing cards obsolete if they were unable to support the extended specification. Furthermore, it would also complicate extensions from the initial demonstration to other Healthcard Systems as these would need to precisely match the CardLink-DiabCard interoperable data set.

8.1.3 Representation of data

The same information can be represented in different ways. Our concern is with the representation of information at the Healthcard Server API. The Healthcard Server must map the data from the structure held on the card to the form of representation agreed at the Healthcard Server API. The possible differences between the representation of information on a particular card and its representation at the Healthcard Server API are noted below.

Logical and physical files

The initial proposals for healthcard interoperability recognise three logical sets of data that can be requested at the Healthcard Server API:

- Card data;
- Administrative data;
- Emergency clinical data.

Each of these sets of data is specified as though it was a logical data file.

The *Card data* needs to be represented by a single physical file. Any Healthcard Server must be able to locate and read this file from any healthcard that supports the operating system(s) with which it can communicate. This is essential since, until this file has been read, the issuing Healthcard System is unknown and therefore the Healthcard Server required to read other data cannot be identified and loaded.

The other two logical files need not be represented as physical files with the format and structure specified at the API. Once the appropriate Healthcard Server has been loaded it should manage mapping between the representation of this information in the card and the representation of the same information at the Healthcard Server API. However, it is self evident that the more closely the healthcard data is aligned with the interoperable data structure at the Healthcard Server API the less processing will be required when this data is read from a card.

Content and detail

The content and detail of the data on the card need not completely match that of the interoperable data set. The differences may include the following:

- The card may contain more information or more detail than is needed to support the interoperable aspects of the Healthcard Server API.
  - When the Healthcard Server receives a command to read interoperable data from the card, it ignores the information or details that are not specified by the Healthcard Server API.
    - WARNING: If any additional detail supported by the card may serve to negate the less detailed information on the card, care must be taken that this is not overlooked when the Healthcard Server selects the information to be presented to the Healthcard Server API.
- The card may not support storage of some information that is optional at the interoperable Healthcard Server API.
  - In this case, the Healthcard Server informs the Healthcard Client Application (via the Healthcard Server API) that the card does not support that item.

- The card may not contain some information, although it is supported by the card, because the information has not been added in respect of a particular patient.
  - In this case the Healthcard Server informs the Healthcard Client Application (via the Healthcard Server API) that the card does not contain the information.
    - The last two cases differ from one another and from explicit statements on the card that something is true, untrue or unknown.

Structure and coding

The card may contain information that is structured or coded in a different way from the structure and coding specified at Healthcard Server API. The differences may include the following:

- The data on the card may be represented as a set of binary flags, enumerated lists or may be compressed to conserve space. At the Healthcard Server the form of representation needs to be specified in a common manner.
  - The most appropriate generalised form of representation at the Healthcard Server API is a Tag-Length-Value (TLV) structure. This is the type of approach mandated by ENV12018 and a variant of the approach was used in CardLink 1.
    - The advantage of this approach is that it allows the nature and length of each item of information to be recognised by the Healthcard Client Application rather than depending on a specific sequencing of bytes.
    - The TLV approach depends upon a shared understanding of tags used in the interoperable data set.
- Some of the data on the card may be represented using code values derived from coding schemes. These coding schemes may be local to a particular Healthcard System or may be recognised registered coding schemes (registered under ISO 7826 or ENV1068). At the Healthcard Server API any coded information must be presented using a registered coding recognised for that purpose or using a code list defined in the interoperability specification.
  - If necessary, the Healthcard Server must map between the coding scheme used on the card and the coding scheme at the interoperable Healthcard Server API.
  - It is strongly recommended that the Healthcard Server should present direct textual translations of any clinical codes that are not shared by the card and Healthcard Server API. There is a danger that code translation may reduce specificity.

The Healthcard Client Application user-interface

There may also be similar differences between the representation of information at the Healthcard Client Application user interface and the same information at the Healthcard Server API. Similar considerations apply as those discussed above. However, within the context of the initial interoperability, demonstration user interfaces should be developed to display all the interoperable information. To minimise the need for further manipulation, this information should be presented in a manner that reflects the specification of the Healthcard Server API.

8.1.4 The role of ENV12018 and standardisation work in CEN TC224 WG12

As a formally adopted European Prestandard ENV12018 should be taken as the foundation of the interoperable data set. However, full implementation of ENV12018 is beyond the scope of the

immediate work on interoperability and there are practical obstacles to use of the structures specified in ENV12018.

To meet the perceived interoperability requirement this report defines three logical data sets and ENV12018 appears to support a similar top level structure (see Table 4). However, even at this level, there are differences between the interoperable data requirements and the formal provisions of ENV12018*.

Table 4. Apparent alignment between interoperable data sets and ENV12018

Interoperable data sets	ENV12018 structures
Card Application data	*DeviceData*
Administrative data	*AdminData*
Emergency data	*ClinicalData* • This includes a subset of emergency data

1. The healthcard interoperability approach uses the *Card data* logical file to identify the application so that the appropriate Healthcard Server can be loaded or configured. It is thus essential that this block of data should be defined unambiguously and as simply as possible. The ENV12018 *DeviceData* object:
   - Is not simple because it can contain information related to security provisions which go well beyond the initial requirements for interoperability.
   - Does not include the Card Issuer identification; this is with the patient identifier in *MajorRecordPerson* which is a part of the Administrative data. Instead it contains specific identifiers of the device applications. In many ways, this may be a preferable approach, but it requires recognised identifiers of the application. The CardLink approach, based on the Card Issuer identifier, was accepted by the interoperability group at its last meeting.
2. The *ClinicalData* object in ENV12018 provides a general framework for recording clinical information but its full provisions are more detailed than those identified by EUROCARDS, CardLink and DiabCard. On the other hand the *LimitedEmergencyData* object is too limited as it provides only a bit map for true/false statements about a list of clinical conditions. For healthcard interoperability a middle way is needed that identifies particular items of emergency information that will be present. It should describe the representation of that information at the interoperable Healthcard Server API.

* The comments in this section are based on an initial comparison of ENV12018 with the requirements identified in this report. Comments from those with more detailed understanding of ENV12018 would be welcome and may lead to revisions in this section.

3. The overall presentation of ENV12018 provides a wide range of options for implementation. This is appropriate to an initial formal standard of this kind. However, to achieve functional interoperability it is necessary to agree a profile that places some constraints on the options.

Despite these points, the initial proposal is that ENV12018 is as a basis for the development and definition of an interoperable profile.

- The data elements and application tags used in ENV12018 should be used where a ENV12018 data object meets the needs for healthcard interoperability.
- The interoperable data set will exclude many optional elements specified in ENV12018. In some cases the profile for the same object may differ according to the information recorded within it (e.g. some items of clinical information may be coded, whereas others are coded and dated and others may include numeric information).
- The top level structure for the interoperable data set will consist of the three logical files described earlier in this document, rather than following ENV12018 specification.
- The Card Issuer and patient identifier will be in the *Card data* logical file which will retain a structure similar to that used in CardLink I.
- A facility to explicitly identify the card applications supported by the healthcard will be added to the *Card Data* logical file. This will follow the approach to coded application identifiers recommended by ENV12018. However, this will be an optional element and, if it is absent, the Healthcard Server will determine the card applications supported according to the Card Issuer.

When considering the application of ENV12018 to interoperable healthcards, it is important to note that further work on this area is now proceeding in CEN TC224 WG12. The intention of this work is to produce a formal Standard for representation of the logical data structures of ENV12018 in a manner appropriate to different card technologies used by patients in health applications. At the time of writing these proposals have not been adopted as a Standard or a Prestandard. However, they contain valuable observations that are of direct relevance to the design of data structures on IC healthcards. Furthermore, during the next year a future revision of the proposals may be balloted and formally adopted.

8.2 Healthcard Server API Data Structures

8.2.1 Introduction

This section specifies structure and representation of data at the Interoperable Healthcard Server API. It does **not** specify the structures and representation of data stored on the card. Designers of new healthcard data structures should refer to ISO7816-6 and ENV12018 for advice on Standard tag allocation schemes and structures for storage of data on cards.

The interoperable healthcard data consists of three separately accessible blocks:

- Card Application Data
- Administrative Data
- Emergency Clinical Data

Each of these sets of information can be separately requested using the Healthcard Server API functions *HciReadData* (to initiate reading) and *HciGetData* (to obtain the results of reading). When the results of reading the card are presented to the Healthcard Server API they are in a format that complies with ASN.1 Basic Encoding Rules (BER). Each element is represented by a Tag-Length-Value combination.

8.2.2 Tag allocation

The tabular form presented in Tables 6 to 8, identifies composite tags for data objects that are identified as **groups** of data items, **sub-groups** and elementary tags for individual **data items** within those **groups** or **sub-groups**. An ASN.1 representation of the same data structures is provided in Chapter 8.4.

Each of the tags shown in the tables is one byte long. They include the settings of bits 7 and 8 to indicate context-specific allocation and bit 6 to indicate the composite or elementary form of the contents. Thus tags for composite items are in the range A0 to BF (binary “101xxxxx” and tags for elementary items are in the range 80 to 9F (binary “100xxxxx”).

For convenience of mapping between the ASN.1 representation and the tabular representation the values of the lower five bits in the tag are allocated sequentially irrespective of the value of bit 6. Thus if the third item in a particular context is composite while all the others are elementary, the tags are allocated as shown in Table 5.

Table 5. Mapping of tags for elementary and composite data items

Tag	Bits 7 and 8 (Class) 10= context specific	Bit 6 (Form) 1=Composite 0=Elementaty	Bits 1 to 5 Item number	Tag shown in ASN.1 Representation (see 8.4)
80	10	0	00000	[0]
81	10	0	00001	[1]
A2	10	1	00010	[2]
83	10	0	00011	[3]

All the tags shown are context-specific either within the context of the overall response to the *HciGetData* command or within the context of the composite elements within that response (referred to as **groups** and **sub-groups)**. A tag could also be allocated to each of the three top level composite objects. These tags would need to be formally registered. However, such tags are not essential to the interoperability demonstration as the HciGetData parameters specifically request a particular top level composite object and the buffer length is returned as separate parameter. Thus the structure of data in the buffer is as illustrated in Figure 3.

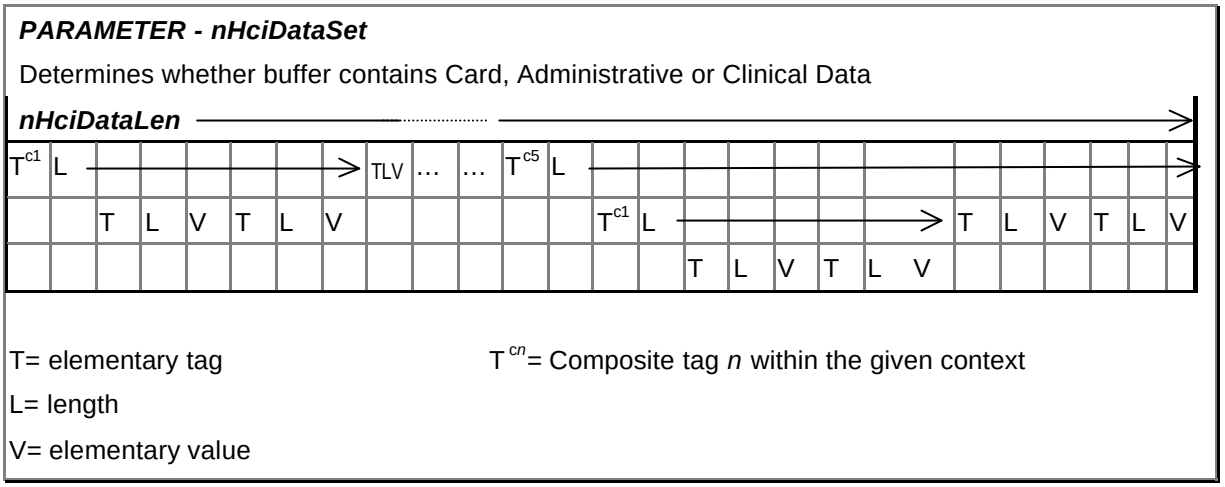

Figure 3. HciGetData parameters and the contents of the buffer

Note

These tag allocation rules are NOT followed if a well established and directly applicable tag for the same data item is already defined. For example, the CardApplicationIdentifier follows the structure of AID from ISO7816-5 and tags for this structure are defined in the standard.

8.2.3 Data types and presentation at the Healthcard Server API

The five elementary data types used at the Healthcard Server API are represented as follows:

- **Octet string** values are consist of an ordered sequence of bytes at the API but may have an a form of visible presentation other than Latin 1. The form of presentation is defined in notes on the specific data item.
- **String** values are presented as octet strings with each octet representing a single character from Latin 1 alphabet (ISO 8859-1).
- **Numeric** data is presented as octet strings with each octet representing a single ASCII character in which the valid characters are digits, the decimal point and the minus sign. Where a mandatory numeric value is unknown or unavailable a zero length is string returned not a string containing the numeric value zero (i.e. not “0” nor “0.0” but “”).
- **Enumerated** values are presented as integers in accordance with ASN.1 Basic Encoding Rules.
- **Dates** are presented as strings of up to eight digits in the format YYYYMMDD. If only the year and month is known a string of six digits is presented YYYYMM and if only the year is

known YYYY. A mandatory date field that is unknown may be presented as a zero length string.

Note

These representations apply only at the Healthcard Server API and do not constrain the use of more compact forms of storage or representation on the card or within the Healthcard Client Application.

8.2.4 Optional and mandatory elements and repetition of elements

For several reasons discussed elsewhere in this report the interoperable data structure needs to allow for the optional inclusion of some data objects. Furthermore, some data objects defined in the specification can be repeated several times. This is represented in the Tables 6 to 8 by the column headed "Occurrences Max, min".

The numbers in this column indicate the minimum and maximum number of occurrences of that object in the data object (**group** or **sub-group**) of which it forms part.

If the minimum number is zero, an object is optional. If the minimum is one or more, an object is mandatory, in each instance of object of which it forms a part. However, if the object of which it forms a part of is optional and is omitted, there will of course be no instances of any of its components.

If the maximum number is one, the object can only occur once in each instance of the object of which it forms a part. If the maximum is greater than one, it can occur up to that number of times in each instance of the object of which it forms a part.

The considerations in this section apply to the Healthcard Server not to the healthcard itself nor to the Healthcard Client Application. Some of the mandatory information can be provided by the Healthcard Server reporting that healthcard does not or cannot hold a particular piece of information (e.g. whether a patient has a particular condition or not).

A more detailed specification of the rules of optionality at different layers in the healthcard interoperability architecture is provided by Tables 13 and 14 in Section 8.3.

8.2.5 Representation of ASN.1 sets and sequences

The items that repeat in the Tables 6 to 8 are represented as sets or sequences in the ASN.1 according to whether the order of multiple instances is significant. The way in which these are represented in the HS-API data structure should accord with theASN.1 Basic Encoding Rules specified by ISO 8825. To retain clarity for users reviewing the data set the tabular representation do not show the additional tags allocated to these repeating elements in accordance with the encoding rules. Technical readers should refer to the ASN.1 representation in 8.4 and to the relevant sections an annexes of ISO 8825.

8.2.6 Representation of clinical information

Several elements of the emergency clinical data specified in Table 8 use a repeating general structure for conveying similar types of information. This structure is a subset of the clinical coded data structure of ENV12018. To understand how this implements previously agreed minimum data sets[3]

[3] Previously agreed administrative and emergency datasets supported by this structure include those of the European Emergency Card, Eurocards WG4, CardLink 1 and DiabCard 2.

refer to Tables 9 to 11. These tables list the codes to be used to represent emergency clinical categories and indicate which of these are mandatory in the interoperability demonstration.

8.2.7 Card Application Data

Purpose

Card Application Data contains the minimum set of essential information about the healthcard that should be free-reading in any interoperable healthcard. Its primary purpose is to identify the card issuer and the method by which other interoperable healthcard data may be read from the card.

Form

This data exists in a file that is directly accessible to any Healthcard Server that can read cards with the same card operating system. This file must include as a minimum the elements described in this document and each such element must be tagged within the file in accordance with this specification. Furthermore, the location and method of access to the Card Application Data file must be publicly specified either by the designer of the first interoperable healthcard that uses cards with a particular operating system. Thereafter, other healthcard designers using cards which use the same operating system must provide a Card Application Data file that is accessible in precisely the same way.

Usage

When a card is inserted it is first reset. In response to the reset the card returns the ATR. This is used to determine the card operating system. A Healthcard Server appropriate to that card operating system is loaded and/or configured[4]. The Card Application Data is then read and the Card Issuer and/or Card Application Identifier is used to determine the method of access to the remaining card data.

Notes on the CardApplicationIdentifier

The CardApplicationIdentifier which is identical in structure with the Application Identifier (AID) specified in ISO7816-5. It consists of up to sixteen bytes. For display purposes and in the configuration file it is represented at the Healthcard Server API as a string of up to 32 hexadecimal characters (each character representing 4 bits).

For the purposes of healthcard interoperability, the CardApplicationIdentifier is returned as part of CardApplicationData. However, the Healthcard Server may obtain this information from one of several alternative sources.

- From the ATR historical bytes (of a single application card)
- From an ATR file or DIR file where the cards support these structures in compliance with ISO7816-4 and ISO7816-5.
- From a separate CardApplicationData file accessible by Healthcard Servers written for this type of card.
- From a built in table that generates an appropriate identifier after recognising the cards from other proprietary characteristics.

Cards that do not store the AID as specified in ISO7816-5 may still be compatible with the healthcard interoperability proposals in this report. Provided they can be recognised from their ATR or by the trial and error approach outlined in 3.2.1, a Healthcard Server can generate AID(s) for any card(s) that it can recognise.

[4] See 4.1 for an explanation of the technique by which the appropriate Healthcard Server is identified.

Table 6. Card Application Data

Name	Tag Hex	Data Type	Max Len (bytes)	Occurrences Min, max	Notes
Card Issuer Identifier	**A0**	**Group**		**1,1**	
> Major Industry Identifier	80	Number	2 Fixed	1,1	"80" for healthcare
> Country Code	81	Number	3 Fixed	1,1	According to EN23166:1994
> Issuer Identifier	82	Number	8	1,1	Allocated by nationally appointed registration authority. In Europe a fixed length of 5 characters has been accepted (ENV12018 and EN1387). However, the proposed ANSI standard on which US numbering will be based has 8 characters plus a check digit. Therefore, European implementers should be aware of the need to handle this longer form.
> Check Digit	83	Number	1 Fixed	1,1	Luhn modulus ten double-add-double check digit of previous three items.
Card Holder Identifier	**81**	**String**	**21**	**0,1**	Unique identifier of the card holder by the Card Issuer. The use of this is optional and at the discretion of the Card Issuer. Where law and ethical constraints permit this may be used for a national heath/insurance identifier.
Card Identifier	**82**	**String**	**28**	**1,1**	Unique identifier of the card by the Card Issuer. The number should be unique within the context of the Card Issuer Identifier. May be a built in card serial number or a concatenation of card holder identifier and an card issue number.
Card Status	**83**	**Enumerated**	**1**	**1,1**	0=Unknown, 1=Test (i.e. not valid for normal use), 2=Normal.

Continued on next page

Name	Tag Hex	Data Type	Max Len (bytes)	Occurrences Min, max	Notes
Card Application Identification (ISO7816-5 Application Template)	**61**	**Group**		**1*,9**	Each instance specifies an application supported by the card. * Cards issued prior to the release of this specification may not include this data item. In this case if the appropriate Healthcard Server must recognise the card by its ATR, Issuer or other characteristics and should then generate an appropriate (agreed or registered) CardApplicationIdentification.
> Card Application Identifier (ISO7816-5 Application Identifier)	4F	Octet-String	16	1,1	An Identifier that follows the structure specified by ISO7816-5. Individual sites are encouraged to register their healthcard applications under ISO7816-5 either at a national or international level. Until Nationally or Internationally registered identifiers are issued the following proprietary identifiers shall be returned. "F0 80 00 01xx" for DiabCard and "F0 80 00 02 xx" for CardLink cards that support healthcard interoperability. In both cases xx is replaced by two characters that define local variants with different Healthcard Server requirements (e.g. "F0 80 00 01 01" EHB CardLink, "F0 80 00 01 05" Santal CardLink, etc.).
> Discretionary Application Data (ISO7816-5 Discretion Data)	*73*	*Sub-group*		*1,1*	*The additional data required to identify healthcard functionality and version numbers should use the ISO7816-5 object "Discretionary data"*
>> Card Application Type	80	Enumerated	1 Fixed	1,1	Specifies the interoperable healthcare application type(s) for this identifier. 0= Administrative and Emergency Clinical, 1= Administrative, 2= Emergency Clinical, 3-8= Reserved for future use, 9= Local use.
>> Card Application Version	81	Number	2 Fixed	1,1	A version number allocated by the organisation that specified the healthcard structure for a particular revision of the Card Application.

8.2.8 Administrative Data

Table 7. Administrative data items required at the Healthcard Server API

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Patient Identification	**A0**	**Group**		**1,3**	**Alternative identifier(s) for the same patient issued by different issuers or naming authorities.**
> Issuer of Patient Identifier	*A0*	*Sub-group*		*1,1*	
>> Major Industry Identifier	80	Number	2 fixed	1,1	"80" for healthcare
>> Country Code	81	Number	3 fixed	1,1	According to EN23166:1994
>> Issuer Identifier	82	Number	8	1,1	Allocated by nationally appointed registration authority. In Europe a fixed length of 5 characters has been accepted (ENV12018 and EN1387). However, the proposed ANSI standard on which US numbering will be based has 8 characters plus a check digit. Therefore, European implementers should be aware of the need to handle this longer form.
>> Check Digit	83	Number	1 fixed	1,1	Luhn modulus ten double-add-double check digit of previous three items.
> Patient Identifier	81	String	21	1,1	The identifier by which the issuer refers to the patient who holds this card.
Name Details	**A1**	**Group**		**1,1**	
> Title	80	String	7	0,1	E.g. "Mr", "Prof.", "Madame"
> Surname prefix	81	String	15	0,1	A prefix to the surname that is typically omitted when calculating alphabetical order (e.g. "von", "van der", "de la")
> Surname	82	String	27	1,1	
> Alternative Surname	83	String	27	0,1	May also be used for second part of double surnames
> Surname suffix	84	String	15	0,1	A suffix to the surname used as a form of address (e.g. "Senior", "the III").
> Forenames	85	String	16	1,3	May hold initials if forenames not specified.
> Preferred forename	86	String	16	0,1	The preferred forename. If not specified the first instance of forenames specifies the preferred forename.
> Surname at Birth	87	String	27	0,1	Maiden name. In case of adopted persons this may be strictly the name at adoption rather than at birth subject to national conventions.

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Language Details	**A2**	**Group**		**0,4**	Where more than one language is specified the languages should usually be presented in order of patient preference.
> Language	80	String	2	1,1	Coded in accordance with ISO 639:1988.
> Ability in language	81	Enumerated	1	0,1	If more than one language is spoken then this proficiency value is used to distinguish between languages in order of preference. 0=Preferred, 1=fluent, 2=fair, 3= poor.
Birth Details	**A3**	**Group**		**1,1**	
> Date of Birth	80	Date	8	1,1	In full YYYYMMDD, year and month YYYYMM or year only YYYY.
> Sex	81	Enumerated	1	1,1	0=Unknown, 1=Male, 2=Female 3 or 9=Other (usually "other" is represented as 9 as specified by ISO 5218:1977. However, the value 3 should also be treated as other to allow 2 bit storage in the card to be used without further mapping).
> Country of Birth	82	Number	3 fixed	0,1	According to EN23166:1994.
Address Details	**A4**	**Group**		**0,2**	**Each instance specifies an address relevant to the card holder.**
> Address Status	80	Enumerated	1	1,1	An identifier of the status and purpose of the address: 0=Current home address of patient, 1=Previous home address of patient.
> Address Structure	*A1*	*Sub-group*		*0,1*	
>> Address Text	80	String	35	1,5	Text of the postal address. Each instance contains one line of text. Address Text should include the postcode even if this is also specified separately.
>> Address Postcode	81	String	8	0,1	The postal code associated with the address.
>> Address Country	82	Number	3 fixed	0,1	The country of the address. According to EN23166:1994.
> Telecom Structure	*A2*	*Sub-group*		*0,1*	
>> Telephone number	80	Number	16	0,3	Complete number including country and area code with no separators. If more than one number is given they should be in the order in which they should be tried when attempting to contact the patient.
>> Facsimile number	81	Number	16	0,1	Complete number including country and area code with no separators.
>> Network Address	82	String	64	0,1	

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Contact Details	**A5**	**Group**		**0,3**	**Each instance specifies an address relevant to the card holder.**
> Contact Name	80	String	30	1,1	The unstructured name of the next of kin or contact person
> Contact Relationship	81	String	16	0,1	The relationship of the next of kin or contact person to the card holder.
> Contact Address Structure	*A2*	*Sub-group*		*0,1*	
>> Contact Address Text	80	String	35	1,5	Text of the postal address. Each instance contains one line of text. Address Text should include the postcode even if this is also specified separately.
>> Contact Address Postcode	81	String	8	0,1	The postal code associated with the address.
>> Contact Address Country	82	Number	3 fixed	0,1	The country of the address. According to EN23166:1994.
> Contact Telecom Structure	*A3*	*Sub-group*		*0,1*	
>> Contact Telephone number	80	Number	16	0,3	Complete number including country and area code with no separators. If more than one number is given they should be in the order in which they should be tried when attempting to contact the patient.
>> Contact Facsimile number	81	Number	16	0,1	Complete number including country and area code with no separators.
>> Contact Network Address	82	String	64	0,1	
Insuring Body Details	**A6**	**Group**		**0,3**	
> Insuring Body Country	80	Number	3 fixed	0,1	The country of the insuring body. According to EN23166:1994.
> Insuring Body Identifier	81	Number	9	1,1	Number identifying the insuring body
> Insuring Body Name	82	String	20	0,1	
> Insuring Body Address Structure	*A3*	*Sub-group*		*0,1*	
>> Insuring Body Address Text	80	String	35	1,5	Text of the postal address. Each instance contains one line of text. Address Text should include the postcode even if this is also specified separately.
>> Insuring Body Address Postcode	81	String	8	0,1	The postal code associated with the address.
>> Insuring Body Address Country	82	Number	3 fixed	0,1	The country of the address. According to EN23166:1994.
> Insuring Body Telecom Structure	*A4*	*Sub-group*		*0,1*	
>> Insuring Body Telephone number	80	Number	16	0,3	Complete number including country and area code with no separators. If more than one number is given they should be in the order in which they should be tried when attempting to contact the patient.
>> Insuring Body Facsimile number	81	Number	16	0,1	Complete number including country and area code with no separators.
>> Insuring Body Network Address	82	String	64	0,1	
>E111 Certificate	*A5*	*Sub-group*		*0,1*	

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
>>Expiry date	80	Date	8	1,1	Expiry date of current E111 certificate.
>Insurance Numbers	*A6*	*Sub-Group*		*1,1*	*Reference number issued or recognised by the insuring body for the purpose of identifying the policy and/or the insured person.*
>>Insured Person Policy Number	80	String	21	0*,1	* Must be present if National Insurance Number is absent.
>>National Insurance Number	81	String	21	0*,1	* Must be present if Policy Number is absent.
>Insured Person	*A7*	*Sub-Group*		*0,1*	*If the card holder is covered by the insurance policy held by another person (e.g. a relative), this sub-group identifies the insured person or main contract holder with the specified insuring body.*
>>Relationship to Patient	80	String	16	0,1	The relationship of the insured person to the card holder.
>>Insured Person Surname	81	String	27	1,1	
>>Insured Person Alternative Surname	82	String	27	0,1	Second surname if required.
>>Insured Person Forenames	83	String	16	1,3	May hold initials if forenames not specified.
>> Insured Person Address Structure	*A4*	*Sub-sub-group*		*0,1*	
>>> Insured Person Address Text	80	String	35	1,5	Text of the postal address. Each instance contains one line of text. Address Text should include the postcode even if this is also specified separately.
>>> Insured Person Address Postcode	81	String	8	0,1	The postal code associated with the address.
>>> Insured Person Address Country	82	Number	3 fixed	0,1	The country of the address. According to EN23166:1994.
>> Insured Person Telecom Structure	*A5*	*Sub-sub-group*		*0,1*	
>>> Insured Person Telephone number	80	Number	16	0,3	Complete number including country and area code with no separators. If more than one number is given they should be in the order in which they should be tried when attempting to contact the patient.
>>> Insured Person Facsimile number	81	Number	16	0,1	Complete number including country and area code with no separators.
>>> Insured Person Network Address	82	String	64	0,1	

8.2.9 Clinical Data

Table 8. Clinical data items required at the Healthcard Server API

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Coded Clinical Details	**A0**	**Group**		**N*,99**	***In practice a the minimum number of occurrences is determined by the list of diseases to be included in the emergency data set. For some of these the Clinical Indicator may be 9 (i.e. Not supported by card).**
> Clinical Emergency Category	80	Number	2	1,1	A number identifying key items of emergency clinical data. The category numbers are specified in this document (see Table 9).
> Clinical Indicator	81	Enumerated	1	1,1	0= Disease or condition indicated by Code recorded as **Absent**. 1= Disease or condition indicated by Code recorded as **Present**. 2= Disease or condition indicated by Code recorded as **Possible** (may be used to indicate uncertainty of the person recording). 8= Supported by card but **not recorded** for this patient. 9= **Not supported** by card (i.e. there is no way that the card could indicate the presence or absence of the condition).
> Clinical coding structure	*A2*	*Sub-group*		*0,1*	*A structure containing more detailed coded information*
>> Coding Scheme Identifier	80	String	6	1,1	Designates the coding scheme from which Clinical Code is derived. Uses identifiers registered in ISO 7826.
>> Clinical Code	81	String	8	1,1	Code used for representation of clinical data.
> Clinical Date	83	Date	8	0,1	Optional date of diagnosis or first occurrence. Full date YYYYMMDD, year & month YYYYMM or year only YYYY.
> Clinical Text	84	String	80	0,1	Optional free text associated with the coded information.
> Clinical Entry Date	85	Date	8	0,1	Date on which this entry was made (or changed). As full date YYYYMMDD.
> Clinical Author	*A6*	*Sub-group*		*0,1*	*The person responsible for recording this clinical item.*
>> Author Country	80	Number	3 fixed	0,1	The country of the author. According to EN23166:1994.
>> Author Identifier	81	Number	12	0,1	Identifier of the person responsible for the entry or change. The identification must be either globally or nationally unique.
>> Author Name	82	String	20	0,1	Optional plain text name of the responsible person.

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Blood Group and Transfusion Details	**A1**	**Group**		**1*,1**	* If no record is present on the card this should still be presented at the Healthcard Server API with Blood Group "U" and (unknown)Blood Transfusion Indicator 9 (e.g. Not supported)
> Blood Group	*A0*	*Sub-group*		*1,1*	
>>ABO Blood Group	80	String	2	1,1	"O", "A", "B", "AB" or "U" = Unknown.
>>Rhesus Factor	81	String	1	1,1	"+" or "–" or "U" = Unknown.
>> Date of Last Blood Grouping	82	Date	8	0,1	Full date YYYYMMDD, year & month YYYYMM or year only YYYY.
>> Blood Grouping Text	83	String	30	0,1	Optional free text description of any additional grouping factors.
> Blood Transfusion	*A1*	*Sub-group*		*1*,1*	** If no record is present on the card this should still be presented at the Healthcard Server API with the Indicator 9 (e.g. Not supported)*
>> Blood Transfusion Indicator	80	Enumerated	1	1,1	0= Blood transfusion recorded as **Never**. 1= Blood transfusion recorded as **Once or more than once**. 2= Blood transfusion recorded as **Unknown**. 8= Supported by card but **not recorded** for this patient. 9= **Not supported** by card (i.e. there is no way that the card could indicate that the patient has or has not had a transfusion).
>> Last Blood Transfusion Date	81	Date	8	0,1	Full date YYYYMMDD, year & month YYYYMM or year only YYYY.
> Blood Group Entry Date	82	Date	8	0,1	Date on which this entry was made (or changed). As full date YYYYMMDD.
> Blood Group Author	*A3*	*Sub-group*		*0,1*	*The person responsible for recording the blood transfusion details.*
>> Author Country	80	Number	3 fixed	0,1	The country of the author. According to EN23166:1994.
>> Author Identifier	81	Number	12	0,1	Identifier of the person responsible for the entry or change. The identification must be either globally or nationally unique.
>> Author Name	82	String	20	0,1	Optional plain text name of the responsible person.

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Immunisation Details	**A2**	**Group**		**0,10**	
> Immunisation Emergency Category	80	Number	2	1,1	Immunisation identifier. The proposed source of this code list is the Immunisation Table in the CardLink specification. See Table 10.
> Immunisation Indicator	81	Enumerated	1	1,1	0= Immunisation recorded as **Never** done. 1= Immunisation recorded as done **at least Once**. 2= Immunisation recorded as **Unknown**. 4= Immunisation recorded as **Adverse reaction**. 8= Supported by card but **not recorded** for this patient. 9= **Not supported** by card (i.e. there is no way that the card could indicate that the patient has or has not had this immunisation).
> Immunisation Status	82	Enumerated	1	0,1	Valid only if indicator is 1. 0= **Unspecified** dose. 1= **First dose** of course. 2= **Second dose** of course. 3= **Third dose** of course. 4= **Completed** course 5= **Booster** dose 9= **Not supported** by card (i.e. there is no way that the card could indicate which injection in the course was given).
> Last Date Immunised	83	Date	8	0,1	The last recorded date of this immunisation. Full date YYYYMMDD, year & month YYYYMM or year only YYYY.
> Immunisation coding structure	*A4*	*Sub-group*		*0,1*	*A structure containing more detailed coded information about an immunisation.*
>> Coding Scheme Identifier	80	String	6	1,1	Designates the coding scheme from which immunisation code is derived. Uses identifiers registered in ISO 7826.
>> Clinical Code	81	String	8	1,1	Code used to specify the immunisation in more detail.
> Immunisation Text	85	String	30	0,1	Optional free text description of any other immunisation.
> Immunisation Entry Date	86	Date	8	0,1	Date on which this entry was made (or changed). As full date YYYYMMDD.
> Immunisation Author	*A7*	*Sub-group*		*0,1*	*The person responsible for recording this immunisation.*
>> Author Country	80	Number	3 fixed	0,1	The country of the author. According to EN23166:1994.
>> Author Identifier	81	Number	12	0,1	Identifier of the person responsible for the entry or change. The identification must be either globally or nationally unique.
>> Author Name	82	String	20	0,1	Optional plain text name of the responsible person.

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Medication Details	**A3**	**Group**		**N*,30**	*In practice a the minimum number of occurrences is determined by the list of drug groups to be included in the emergency data set. For some of these the Medication Indicator may be 9 (i.e. Not supported by card).
>Medication Emergency Category	80	Number	2	1,1	An identifier of drug categories that enables the emergency information to be conveyed without need for more detailed drug information. See Table 11
> Medication Indicator	81	Enumerated	1	1,1	0= Medication with drug in this group recorded as **Absent**. 1= Medication with **at least One** drug in this group recorded. 2= Medication with drugs in this group recorded as **Unknown**. 5= **Past or short-term** medication with at least one drug in this group recorded but not part of regular current medication. 6= **Intermittent** medication with at least one drug in this group is recorded (e.g. anti-histamines for hayfever only taken at some times in the year). 8= Supported by card but **not recorded** for this patient. 9= **Not supported** by card (i.e. there is no way that the card could indicate the presence or absence of the condition).
> Medication coding structure	*A2*	*Sub-group*		*0,6*	*A structure containing more detailed coded information*
>> Coding Scheme Identifier	80	String	6	1,1	Designates the coding scheme from which Medication Code is derived. Uses identifiers registered in ISO 7826.
>> Medication Code	81	String	8	1,1	Code(s) for medication item. In the case of ATC codes several codes may apply to the same item representing different ingredients in this case the coding structure repeats
> Medication Drug Name	83	String	50	0,1	Optional textual representation of a particular drug name. Ideally this should conform to the International Non-proprietary Name (INN). Only valid if the Medication Indicator is 1, 5 or 6.
> Medication Dosage Code	84	String	2	0,4	Optional coded representation of dosage. Codes according to Table 12 represented as a sequence that constructs a dosage instructions.
> Medication Dosage	85	String	50	0,1	Optional textual summary of dosage instructions.
> Medication Started Date	86	Date	8	0,1	Date on which this medication was started. Valid if the Medication Indicator is 1, 5 or 6. Full date YYYYMMDD, year & month YYYYMM or year only YYYY.
> Medication Ended Date	87	Date	8	0,1	Date on which this medication was stopped. Only valid if the Medication Indicator is 5 (i.e. Past of Short-term). Full date YYYYMMDD, year & month YYYYMM or year only YYYY.

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
> Medication Entry Date	88	Date	8	0,1	Date on which this entry was made (or changed). As full date YYYYMMDD.
> Medication Author	*A9*	*Sub-group*		*0,1*	*The person responsible for recording this medication item.*
>> Author Country	80	Number	3 fixed	0,1	The country of the author. According to EN23166:1994.
>> Author Identifier	81	Number	12	0,1	Identifier of the person responsible for the entry or change. The identification must be either globally or nationally unique.
>> Author Name	82	String	20	0,1	Optional plain text name of the responsible person.
Clinical Address Details	**A4**	**Group**		**0,9**	**Each instance specifies an address at which clinical information relevant to the card holder may be available.**
> Clinical Address Name	80	String	30	1,1	The unstructured name of the person or clinic
> Clinical Address Relationship	81	String	16	0,1	The clinical relationship to the patient
> Clinical Address Structure	*A2*	*Sub-group*		*0,1*	
>> Clinical Address Text	80	String	35	1,5	Text of the postal address. Each instance contains one line of text. Address Text should include the postcode even if this is also specified separately.
>> Clinical Address Postcode	81	String	8	0,1	The postal code associated with the address.
>> Clinical Address Country	82	Number	3 fixed	0,1	The country of the address. According to EN23166:1994.
> Clinical Telecom Structure	*A3*	*Sub-group*		*0,1*	
>> Clinical Telephone number	80	Number	16	0,3	Complete number including country and area code with no separators. If more than one number is given they should be in the order in which they should be tried when attempting to contact the patient.
>> Clinical Facsimile number	81	Number	16	0,1	Complete number including country and area code with no separators.
>> Clinical Network Address	82	String	64	0,1	
Optical Prescription Details	**A5**	**Group**		**0,1**	
> Optical prescription	80	**String**	**40**	**1,1**	Summary of optical prescription to speed issuing of new glasses or contact lenses.
> Optical Prescription Date	81	Date	8	0,1	Optional date on which this optical prescription was last updated. Full date YYYYMMDD, year & month YYYYMM or year only YYYY.

Name	Tag	Data Type	Max Len (bytes)	Occurrences Min, max	Note
Update Details	**A6**	**Group**		**1,1**	**Information about the last update of the Clinical Data.**
> Date of Last Clinical Update	80	Date	8	1,1	Date on which the Clinical Data on the card was last updated. Only Full date YYYYMMDD is applicable.
> Responsible Party	*A1*	*Sub-group*		*0,1*	*The person or organisation responsible for the last update of the card.*
>> Responsible Party Country	80	Number	3 fixed	0,1	The country of the responsible party. According to EN23166:1994.
>> Responsible Party Identifier	81	Number	12	0,1	Identifier of the person or organisation responsible for the update. The identification must be either globally or nationally unique.
>> Responsible Party Name	82	String	20	0,1	Optional plain text name of the responsible person.

Responsibility for interoperable healthcard data

The update details are informative only and do not provide a guarantee of authorship of particular items on the card. Future revisions of this specification will need to address this topic in association with the development of interoperable healthcare professional cards. Until a globally acceptable mechanism for recording authentication is developed there is little point in attempting to represent a digital signature in relation to either individual items or the card as a whole. Therefore, all users of interoperable healthcards must be made fully aware that information on the card should be treated as unconfirmed. Whenever possible, they should verify information on the card with the patient, and with healthcare professionals who have treated the patient, before being used as a basis for safety critical decisions[5].

[5] Clinical Address Details held on the card are presented at the Healthcard Server API to facilitate checking.

8.2.10 Clinical information items - coded representation and minimum requirements

Table 8 specifies the general structure of clinical information required at the API. The following tables provide the short code lists for the Emergency Categories that are to be used at the interoperable Healthcard Server API. Where the codes are marked as mandatory (M) the Healthcard Server is obliged to report an item with this category code. However, the associated Indicator field can be set to a value that indicates that the information is not recorded on the card or is not supported by the card.

Note

These code lists may be extended in future versions of this specification but values may not be reused. Local definition of additional values is forbidden as it may have serious consequences when cards are read at alternate sites. Formal procedures to register these code lists under ISO7826 are proposed. Agreement is needed on the appointment of an Issuing Organisation.

Table 9. Codes for Clinical Emergency Category at the Healthcard Server API

Meaning	Clinical Emergency Category	Code ICD9.CM	Mandatory or Optional	Notes
Diseases				
Asthma	01	493	M	
Heart Disease	02		M	Including congenital, ischaemic and other heart diseases.
Cardiovascular Disease	03		O	See also more specific item Heart Disease.
Epilepsy	04	345	M	Including any types of fit.
Neurological disorders	05		O	See also more specific items epilepsy.
Coagulation deficiency	06	286	M	Including haemophilia.
Diabetes	07	250	M	
Glaucoma	08	365	M	
Other significant diseases	00		O	Valid only if specified in Clinical Text
Procedures				
Dialysis Treatment	31		M	
Removal of an Organ	32		M	
Transplanted Organ	33		M	
Removable Prosthesis	34		M	
Pacemaker	35		M	
Other Procedures	30		O	Valid only if specified in Clinical Text
Allergies				Serious allergies should always be stored on the card but it is not mandatory to report the absence of these allergies at the Healthcard Server API
Analgesics	71		O	
Animal hair	72		O	
Antibiotics	73		O	
Citrus fruits	74		O	
Dust (or dust mite)	75		O	
Eggs	76		O	
Fish or Shellfish	77		O	
Iodine	78		O	
Milk	79		O	
Nuts	80		O	
Pollen	81		O	
Other Allergies	70		O	Valid only if specified in Clinical Text

Table 10. Codes for Immunisation Emergency Category at the Healthcard Server API

Meaning	Immunisation Emergency Category	Mandatory or Optional	Notes
Anthrax	01	O	
BCG	02	O	
Cholera	03	O	
Diphtheria	04	O	
Diphtheria, Pertussis & Tetanus	05	O	
Diphtheria & Tetanus	06	O	
Haemophilus Influenza B	07	O	
Hepatitis A	08	O	
Hepatitis B	09	O	
Influenza	10	O	
Japanese encephalitis	11	O	
Measles	12	O	
Measles, Mumps and Rubella	13	O	
Measles & Rubella	14	O	
Meningococcal Infection (A&C)	15	O	
Mumps	16	O	
Pertussis	17	O	
Pneumococcus	18	O	
Polio (inactivated vaccine)	19	O	
Polio (oral vaccine)	20	O	
Rabies	21	O	
Rubella	22	O	
Tetanus	23	O	
Tick Borne Encephalitis	24	O	
Typhoid (oral)	25	O	
Typhoid (injection)	26	O	
Yellow Fever	27	O	
Others	00	O	Valid only if specified by Immunisation Text.

Table 11. Codes for Medication Emergency Category at the Healthcard Server API

Meaning	Medication Emergency Category	Mandatory or Optional	Notes
Anti-arrhythmic	01	M	See also more specific item for digitalis.
Anti-coagulants	02	M	
Anti-convulsants	03	M	
Anti-diabetics	04	M	See also more specific item for insulin.
Anti-histamines	05	M	
Anti-hypertensives	06	M	See also more specific items such as Beta-blockers and Diuretics.
Beta blockers	07	M	
Corticosteroids	08	M	
Cytostatics & cytotoxics	09	M	
Digitalis	10	M	
Diuretics	11	M	
Insulin	12	M	
Monoamine oxidase inhibitors	13	M	
Psycholeptics	14	M	
Others	00	O	Valid only if Drug Name is specified.

8.2.11 Drug dosage codes

Table 12 is a code list proposed by the CardLink 1 Eastern Health Board site as a way of representing drug dosage in a language independent form. There is no known formal standard for this although many proprietary systems use different schemes. Up to four of these codes can be used to express a dosage instruction.

The drug dosage code data item is at present optional and is meaningless unless the drug is identified using either the Medication Coding Structure(s) or Medication Drug Name.

It is not intended that the drug dosage code should be used by an interoperable system for any purpose other than to inform a healthcare professional of the patients current drug regime. In particular, the representation of medication and dosage described in this specification is **not** intended to act as an electronic prescription nor as instructions to administer drugs in accord with a specified regime.

A comparison of these codes with another system for coding dosage is provided in Annex B.

Table 12. Optional codes to be used for drug dosage at the Healthcard Server API

Dosage code	Meaning
&	and
-1	then reduce by one each day until the course is complete
.5	.take half a 5ml spoonful
.7	.take three quarters of a 5ml spoonful
05	.take one 5ml spoonful
10	.take two 5ml spoonfuls
12	.one or two to be taken
14	for 14 days
15	.take three 5ml spoonfuls
1B	.one to be taken twice daily
1D	.one to be taken daily
1H	every hour
1M	one in the morning
1N	one at night
1P	.one puff to be inhaled
1Q	.one to be taken four times daily
1S	.take one to start then
1T	.one to be taken three times daily
1U	.one to be used
20	.take four 5ml spoonfuls
23	.two or three to be taken
24	every twenty four (24) hours
2B	.two to be taken twice daily
2D	.two to be taken daily
2H	every two hours
2M	two in the morning
2N	two at night

Dosage code	Meaning
2P	.two puffs to be inhaled
2Q	.two to be taken four times daily
2S	.take two to start then
2T	.two to be taken three times daily
2U	.two to be used
30	.take six 5ml spoonfuls
34	.three or four to be taken
3B	.three to be taken twice daily
3D	.three to be taken daily
3H	every three hours
3M	three in the morning
3N	three at night
3P	three puffs to be inhaled
3Q	.three to be taken four times daily
3S	.three to start then
3T	.three to be taken three times daily
3U	.three to be used
4B	.four to be taken twice daily
4D	.four to be taken daily
4H	every four hours
4N	four at night
4P	.four puffs to be inhaled
4Q	.four to be taken four times daily
4S	.take four to start then
4T	.four to be taken three times daily
4U	.four to be used
5	.take one 5ml spoonful
5D	.five to be taken daily
6D	.six to be taken daily
6H	every six hours
8H	every eight hours
AA	.apply to affected areas
AC	.take half to one hour before food
AD	on alternate days
AF	after food
AP	.apply
AS	.avoid exposure of skin to direct sunlight or sun
AV	.avoid alcoholic drink
BD	twice daily
BE	into both eyes
BR	into both ears
CC	with food
CD	.warning. may cause drowsiness

Dosage code	Meaning
CU	.this medicine may colour the urine or stools
D1	.instil one drop
D2	.instil two drops
D3	.instil three drops
D4	.instil four drops
D5	.instil five drops
D6	.instil six drops
D8	.dissolve under tongue. discard eight weeks after
DA	daily
DC	.may cause drowsiness. may colour urine or stools
DR	.may cause drowsiness. if affected do not drive or
DS	.dissolve
DT	.dissolve under the tongue
EO	the eye ointment
ER	the ear drops
ES	.take an hour before food or on an empty stomach
EV	in the evening
EX	.for external use only
EY	the eye drops
F1	for one day
F2	for two days
F3	for three days
F4	for four days
F5	for five days
F6	for six days
F7	for seven days
FD	.warning, follow the printed instructions with the
FL	.caution flammable: keep away from naked flames
HH	half an hour before food
HT	.half a tablet
I	.insert
I1	.insert one
I2	.insert two
IJ	.for injection only
IN	.for inhalation only
IR	if required
IW	in water
LE	into left eye
LR	into left ear
M3	(maximum of three per day)
M6	(maximum of six per day)
M8	(maximum of eight per day)
MA	in the morning

Dosage code	Meaning
MD	as directed
ML	between meals
MN	morning and night
MT	not more than ___ in 24 hours
MW	not more than ___ in 24 hours or ___ in any one week
NA	.do not take remedies containing aspirin while taking
NI	.do not take iron or indigestion remedies with this
NM	.do not take milk. iron or indigestion remedies with
NO	at night
NR	.do not take indigestion remedies at the same time at
NS	.for nasal use only
NT	.not to be swallowed or taken
OR	or
OS	.use only with spinhaler
PC	.take with or after food
PD	.warning. causes drowsiness which may persist the
PP	when required for pain
PR	as required
PW	.take with plenty of water
QD	four times daily
QH	every four hours
RE	into right eye
RG	.take at regular intervals. complete the course
RO	.use only with rotahaler
RR	into right ear
SB	.shake the bottle well
SC	.for scalp application only
SP	to be applied sparingly
ST	.do not stop taking this medicine except on your
SU	.to be sucked or chewed
SW	.to be swallowed whole. not chewed
T1	.take one
T2	.take two
T3	.take three
T4	.take four
T5	.take five
TD	three times daily
TH	then
TK	.to be taken
TS	teaspoonfuls
U1	.one to be used
U2	.two to be used
U3	.three to be used

Dosage code	Meaning
US	.use
VG	for vaginal use
WA	.dissolve or mix with water before taking

8.3 Optional data at different levels in the Healthcard System

8.3.1 Introduction

In the previous section, Table 6 to Table 11 specified the data structure and representation required at the Healthcard Server API. Those tables specified the optional nature of some data items at this API. However, in practice we must consider what optionality means with regard to the design and use of the card, the interface and the application. Table 13 and Table 14 specify which data items and items of clinical information must be supported at different points in an interoperable Healthcard System.

8.3.2 HS-API optionality

It may be mandatory for an item to be present at the Healthcard Server API. In this case the Healthcard Server may obtain the required information by:

- Reading it from the healthcard
- Deriving it from its knowledge of the healthcard
  - E.g. the coding scheme used to represent a particular type of information on healthcards from a particular issuer may be known to the Healthcard Server and can thus be passed to the API without existing on the card.
- Deriving it from the absence of information on the card
  - E.g. if information is absent from the card it may return an indication that the information is not stored on the card.

It may be optional for an item to be present at the Healthcard Server API because:

- It is only relevant to some instances
  - E.g. a person may not have a phone number.
- It is optional for the information to be made available to interoperable systems
  - E.g. some issuers may restrict access to information that is freely available in other countries.

8.3.3 Card design optionality

It is mandatory for an interoperable healthcard to enable storage of some specified data items. This does not imply that the information must be stored for every patient. However, it does mean that it can be stored if required. Thus an item may be mandatory in the card design and optional at the Healthcard Server API.

Card storage for other items is optional. This allows older less sophisticated cards to coexist with cards that support a wider range of interoperable data. In some cases an item that is mandatory at the Healthcard Server API may be optional in the card design because it can be derived by the Healthcard Server from its knowledge about the card.

The fact that a data item is mandatory in the card design does not mean it must be stored in a particular way. The form in which the information is stored is not mandated provided that it can be mapped unambiguously to the Healthcard Server API structure.

8.3.4 Card user optionality

Some items that can be stored on the card are also mandatory in the sense that the user or issuer of the card must fill in valid data. For example, the patient's name is required to make the card useful from the perspective of interoperability. An item that is mandatory from the card user perspective must be mandatory for card design.

Other items that must be supported by the card design are optional from the perspective of the user or issuer. These items may be left blank or entirely omitted from a particular patient's card. For example, it may be mandatory for the card to enable the patient's blood group to be recorded but the user cannot be compelled to complete this field for every patient. In some cases, the information may be unavailable and in other cases a patient may object to the information being added to the card.

8.3.5 Application visibility

The final aspect of optionality is whether the data passed over the Healthcard Server API must be made visible to the user. It can be argued that the Healthcard Client Application should display all the interoperable information read from the card. However, there are several reasons why this may be impractical or undesirable.

Some data items are considered to be so important that interoperability cannot be achieved unless they are seen by the user. In the case of other data items, the users of the application may not expect or want to see particular data. For example, if the card contains coded clinical information using a national coding scheme there is little point in displaying these codes to the user of a system that does not use this coding scheme. However, interoperable systems that share this coding scheme may benefit from access to this information[6].

Application visibility is mandatory for some data items that are not mandatory for the card design or card user. An example of this is the "Surname prefix" (e.g. "van der", "von", etc). Some cards may absorb these elements with the "Surname". However, a Healthcard Client Application may read an interoperable healthcard in which this element is separated. In this case the Healthcard Client Application must display the full surname including the prefix.

8.3.6 Comparison with CardLink 1 and DiabCard 2 data sets

Table 13 and Table 14 also include columns that provide a comparison with the current CardLink and DiabCard data sets.

The comparison with CardLink confirms that existing CardLink cards meet the mandatory requirements. Some extensions are required to the Healthcard Client Application to support the display of some elements which may be present on other interoperable cards and which if present must be visible to the user. However, the main changes required concern the structure of the Healthcard Server API rather than the content of the emergency and administrative data sets.

The comparison with DiabCard reveals a wider range of differences. This is unsurprising since the DiabCard is a specialty based project which did not originally set out to support a general purpose emergency data set.

---

[6] To cover for this eventuality a simple coding scheme is specified to identify the key emergency items and it is mandatory to display the information represented by these codes. This information may be stored explicitly on the card or derived by mapping from the codes used on the card.

The interoperability packages related to DiabCard 3 and CardLink 2 assume that the content of the interoperable emergency data will be based on the CardLink data set. This specification is fully in line with this principal.

Without a more flexible API data structure, of the type specified, the interoperability specification would become an obstacle to further development.

8.3.7 Key to Table 13 and Table 14.

In these tables there is a column for each of the aspects of interoperability and a column for each of two current data sets.

In the Healthcard Server API columns the notation used in Table 6 to Table 8 is used to denote the minimum and maximum permissible number of occurrences.

The next three columns use the following key

M = Mandatory,

R = Recommended for future mandatory status but optional in initial interoperability demonstration. Items marked R should be included in new cards and applications but may not be available in CardLink and DiabCard.

O = Optional

rM = Mandatory if the recommended object of which it forms part is present

oM = Mandatory if the optional object of which it forms part is present

oR = Recommended if the optional object of which it forms part is present

(*n*) = Reference to note number *n* at the end of the table.

The columns relating to CardLink and DiabCard contains a tick or cross to indicate that the item is or is not covered by the current data sets.

Table 13. Requirements for data elements at different levels in an interoperable card system and a comparison with current data sets

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
Card Issuer Identifier	**1,1**	**M**	**M**	**M (1)**	✓	**û**
> Major Industry Identifier	1,1	M	M	M	✓	û
> Country Code	1,1	M	M	M	✓	û
> Issuer Identifier	1,1	M	M	M	✓	û
> Check Digit	1,1	M	M	M	✓	û
Card Holder Identifier	**0,1**	**R**	**R**	**O**	✓	**ü**
Card Identifier	**1,1**	**R**	**R**	**O**	×/ü from Card Holder ID + version	**ü**
Card Status	**1,1**	**R**	**R**	**M**	× report as unknown	**×**
Card Application Identification	**1,9**	**M(2)**	**M(2)**	**M (2)**	Can be derived from card data.	**û**
> Card Application Identifier	1,1	M	M	M **(2)**	Can be derived from card data.	û
> *Discretionary Application Data*	1,1	M	M	M **(2)**		
> >Card Application Type	1,1	M	M	M **(2)**	Always “0”	
> >Card Application Version	1,1	M	M	M **(2)**	✓	û

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
Patient Identification	**1,3**	**O (3)**	**O (3)**	**M**	✓	**ü**
> Issuer of Patient Identifier	*1,1*	*O*	*O*	*M*	× report as Card Issuer	ü
>> Major Industry Identifier	1,1	O	O	M	× report as 80	û
>> Country Code	1,1	O	O	M	× report as Card Issuer	û
>> Issuer Identifier	1,1	O	O	M	× report as Card Issuer	û
>> Check Digit	1,1	O	O	M	× report as Card Issuer	û
> Patient Identifier	1,1	O	O	M	✓	✓
Name Details	**1,1**	**M**	**M**	**M**	✓	**ü**
> Title	0,1	R	O	O	×	ü
> Surname prefix	0,1	R	O	M (4)	×	û
> Surname	1,1	M	M	M	✓	ü
> Alternative Surname	0,1	R	O	M	✓ follows $ in surname	û
> Surname suffix	0,1	R	O	R (4)	×	û
> Forenames	1,4	M	M	M (5)	✓	ü
> Preferred forename	0,1	O	O	R (5)	✓	ü
> Surname at Birth	0,1	R	O	R	ü maiden name	ü
Language Details	**0,4**	**M**	**O**	**M**	✓	**ü**
> Language	1,1	M	M	M	✓	ü
> Ability in language	0,1	O	O	R	×	û
Birth Details	**1,1**	**M**	**M**	**M**	ü	**ü**
> Date of Birth	1,1	**M**	**M**	**M**	ü	**ü**
> Sex	1,1	**M**	**M**	**M**	ü	**ü**
> Country of Birth	0,1	R	O	O	ü	ü

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
Address Details	**0,2**	**R**	**O**	**R**	ü	**ü**
> Address Status	1,1	rM (6)	oM (6)	rM (6)	ü separate current/previous	ü /û only current
> Address Structure	*0,1*	rM	oM	rM	ü	ü
>> Address Text	1,5	rM (7)	oM	rM (7)	ü	ü structured li1,li2, city
>> Address Postcode	0,1	R	O	rO	û/ü not structured	ü
>> Address Country	0,1	R	O	rM (7)	û/ü not structured	ü/û free text
> Telecom Structure	*0,1*	R	O	oR	û	ü
>> Telephone number	0,3	rM (8)	O	rM (8)	û	ü
>> Facsimile number	0,1	R	O	O	û	û
>> Network Address	0,1	R	O	O	û	û
Contact Details	**0,3**	**R**	**O**	**R**	ü Next of Kin	**û**
> Contact Name	1,1	rM	oM	rM	ü first 2 subfields of NoK	û
> Contact Relationship	0,1	R	O	O	ü third subfield of NoK	û
> Contact Address Structure	*0,1*	rM	oM	rM	ü	û
>> Contact Address Text	1,5	rM (7)	oM	rM (7)	ü	û
>> Contact Address Postcode	0,1	R	O	rM	û/ü not structured	û
>> Contact Address Country	0,1	R	O	rM (7)	û/ü not structured	û
> Contact Telecom Structure	*0,1*	rM	oM	rM	û	û
>> Contact Telephone number	0,3	rM (8)	O	rM (8)	ü final subfield of NoK	û
>> Contact Facsimile number	0,1	R	O	O	û	û
>> Contact Network Address	0,1	R	O	O	û	û
Insuring Body Details	**0,3**	**R**	**O**	**R**	ü	**û**
> Insuring Body Country	0,1	rM	O	rM	ü	û
> Insuring Body Identifier	1,1	rM	oM	rM	ü Paying authority code	û
> Insuring Body Name	0,1	R	O	oR	ü	û
> Insuring Body Address Structure	*0,1*	R	O	O	û	û
>> Insuring Body Address Text	1,5	rM (7)	oM	rM (7)	û	û
>> Insuring Body Address Postcode	0,1	R	O	rM	û	û
>> Insuring Body Address Country	0,1	R	O	rM (7)	û	û
> Insuring Body Telecom Structure	*0,1*	rM	oM	rM	û	û
>> Insuring Body Telephone number	0,3	rM (8)	O	rM (8)	û	û

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
>> Insuring Body Facsimile number	0,1	R	O	O	û	û
>> Insuring Body Network Address	0,1	R	O	O	û	û
>E111 Certificate	*0,1*	*R*	*O*	*R*	ü	û
>>Expiry date	1,1	rM	oM	rM	ü	û
>Insurance Numbers	*1,1*	*rM*	*oM*	*rM*	ü	û
>>Insured Person Policy Number	0,1	R	oM (9)	O	ü subfield 2 of IPD	û
>>National Insurance Number	0,1	R	oM (9)	O	ü subfield 1 of IPD	û
>Insured Person	***0,1***	***R***	***O***	***R***	ü	**û**
>>Relationship to Patient	0,1	rM	O (9)	rM	ü subfield 4 of IPD	û
>>Insured Person Surname	0,1	rM	O (9)	rM	ü subfield 4 of IPD	û
>>Insured Person Alternative Surname	0,1	R	O	R (4)	ü subfield 4 of IPD after $	û
>>Insured Person Forenames	0,1	rM	O (9)	rM	ü subfield 3 of IPD	û
>> Insured Person Address Structure	*0,1*	rM	oM	rM	ü	û
>>> Insured Person Address Text	1,5	rM	oM	rM	ü subfield 5 of IPD	û
>>> Insured Person Address Postcode	0,1	R	O	rM	û/ü not structured	û
>>> Insured Person Address Country	0,1	R	O	rM	û/ü not structured	û
>> Insured Person Telecom Structure	*0,1*	rM	oM	rM	ü	û
>>> Insured Person Telephone number	0,3	rM (8)	O	rM (8)	ü subfield 6 of IPD	û
>>> Insured Person Facsimile number	0,1	R	O	O	û	û
>>> Insured Person Network Address	0,1	R	O	O	û	û

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
Coded Clinical Details	**N*,99**	**M (10)**	**R (10)**	**M (10)**	ü	**ü**
> Clinical Emergency Category	1,1	M (11)	rM (11)	M (11)	ü from tags (see below)	
> Clinical Indicator	1,1	M (12)	rM (12)	M (12)	ü derivable see note 12	ü derivable see note 12
> Clinical coding structure	*0,1*	*R*	*R*	*R*	ü	û
>> Coding Scheme Identifier	1,1	rM	rM	rM	ü derivable for ICD9.CM	û
>> Clinical Code	1,1	rM	rM	rM	ü ICD9.CM	û
> Clinical Date	0,1	R	O	R	û	û
> Clinical Text	0,1	O	O	R	û	û
> Clinical Entry Date	0,1	O	O	O	û	û
> Clinical Author	*0,1*	O	O	O	û	û
>> Author Country	0,1	O	O	O	û	û
>> Author Identifier	0,1	O	O	O	û	û
>> Author Name	0,1	O	O	O	û	û
Blood Group and Transfusion Details	**1*,1**	**M (10)**	**O**	**M (10)**	ü	**ü**
> Blood Group	*1,1*	*M*	*O*	*M*	ü	ü
>>ABO Blood Group	1,1	M	oM	M	ü map from numeric code	ü
>>Rhesus Factor	1,1	M	oM	M	ü map from numeric code	ü
>> Date of Last Blood Grouping	0,1	R	O	R	ü	ü
>> Blood Grouping Text	0,1	R	O	R	û	ü
> Blood Transfusion	*1*,1*	*M*	*O*	*R*	ü	ü
>> Blood Transfusion Indicator	1,1	M (12)	M (12)	M (12)	ü derivable see note 12	ü
>> Last Blood Transfusion Date	0,1	R	R	R	ü	û
> Blood Group Entry Date	0,1	O	O	O	û	û
> Blood Group Author	*0,1*	O	O	O	û	û
>> Author Country	0,1	O	O	O	û	û
>> Author Identifier	0,1	O	O	O	û	û
>> Author Name	0,1	O	O	O	û	û

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
Immunisation Details	**0,9**	**R (10)**	**R (10)**	**R (10)**	ü	**ü**
> Immunisation Emergency Category	1,1	rM (11)	rM (11)	rM (11)	ü first 2 chars of field	? derivable
> Immunisation Indicator	1,1	rM (12)	rM (12)	rM (12)	ü derivable see note 12	?? derivable see note 12
> Immunisation Status	0,1	O	O	O	û	û
> Last Date Immunised	0,1	rM	rM	rM	ü last 6 chars of field	û
> Immunisation coding structure	*0,1*	O	O	O	û	û
>> Coding Scheme Identifier	1,1	oM	oM	oM	û	û
>> Clinical Code	1,1	oM	oM	oM	û	û
> Immunisation Text	0,1	O	O	O	û	ü
> Immunisation Entry Date	0,1	O	O	O	û	û
> Immunisation Author	*0,1*	O	O	O	û	û
>> Author Country	0,1	O	O	O	û	û
>> Author Identifier	0,1	O	O	O	û	û
>> Author Name	0,1	O	O	O	û	û
Medication Details	**N*,19**	**M (10)**	**R (10)**	**M (10)**	ü	**ü**
>Medication Emergency Category	1,1	M (11)	rM (11)	M (11)	ü derive from flag	ü
> Medication Indicator	1,1	M (12)	rM (12)	M (12)	ü derivable see note 12	ü derivable see note 12
> Medication coding structure	*0,6*	*R*	*R*	*R*	ü	û
>> Coding Scheme Identifier	1,1	rM	rM	R	ü derivable for IPU and ATC	û
>> Medication Code	1,1	rM	rM	rM	IPU and ATC codes	û

Name	HS-API occurrences	Card design Optionality	Card user Optionality	Application visibility	CardLink	DiabCard
> Medication Drug Name	0,1	O	O	O	û	û
> Medication Dosage Code	0,1	O	O	O	û	û
> Medication Dosage	0,1	O	O	O	û	û
> Medication Started Date	0,1	O	O	O	û	û
> Medication Ended Date	0,1	O	O	O	û	û
> Medication Entry Date	0,1	O	O	O	û	û
> Medication Author	*0,1*	O	O	O	û	û
>> Author Country	0,1	O	O	O	û	û
>> Author Identifier	0,1	O	O	O	û	û
>> Author Name	0,1	O	O	O	û	û
Clinical Address Details	**0,9**	**R**	**O**	**R**	ü GP +2 others	**ü** GP only
> Clinical Address Name	1,1	rM	oM	rM	ü GP only - first subfield	ü/û GP ID only
> Clinical Address Relationship	0,1	O	O	O	û/ü derivable for GP address	û/ü derivable for GP
> Clinical Address Structure	*0,1*	rM	O	rM	ü GP only	û
>> Clinical Address Text	1,5	rM (7)	oM	rM (7)	ü GP only - second subfield	û
>> Clinical Address Postcode	0,1	R	oR	rM	û/ü not structured	û
>> Clinical Address Country	0,1	R	oR	rM (7)	û/ü not structured	û
> Clinical Telecom Structure	*0,1*	rM	O	rM	ü	û
>> Clinical Telephone number	0,3	rM (8)	O	rM (8)	ü GP only - third subfield	û
>> Clinical Facsimile number	0,1	R	O	O	û	û
>> Clinical Network Address	0,1	R	O	O	ü	û
Optical Prescription Details	**0,1**	**R**	**O**	**O**	ü	**û**
> Optical prescription	**1,1**	**rM**	**O**	**O**	ü	**û**
> Optical Prescription Date	0,1	R	O	O	û	û
Update Details	**1,1**	**M**	**M**	**M**	ü	**û**
> Date of Last Clinical Update	1,1	M	M	M	ü	û
> Responsible Party	*0,1*	*M*	*M*	*M*	ü	û
>> Responsible Party Country	0,1	R	R	R	û/ü not structured	û
>> Responsible Party Identifier	0,1	R	R	M	û/ü not structured	û
>> Responsible Party Name	0,1	R	R	M	ü	û

8.3.8 Notes on Table 13

1. Issuer identifiers need not be visible as numbers in the application if the application can present the country and/or name of the issuer.
2. If the Card Application Identifier is not available on the card, it can be generated by a Healthcard Server that has recognised the card by some other method (e.g. ATR, Card Issuer or trial and error).
3. If there is no separate patient identifier on the card the CardHolderIdentifier is returned as the patient identifier.
4. Surname prefix, surname and surname suffix may be presented in a single field in client applications that do not need to make the distinction between these name elements.
5. An application should as a minimum display the preferred forename or if this is not accessible the first forename. Another alternative is to display the concatenation of all the forenames in a single field.
6. The current address need be supported and displayed for the purposes of interoperability and in some cases this may not be available due to privacy restrictions.
7. The application should be able to display the full address consisting of 5 lines of 35 characters. However, cards that support shorter addresses can still be fully interoperable in read-only mode. It is recommended that that, the application should interpret the country code and display it as text.
8. The application should display telephone numbers in a form that is useful for local dialling. However, at the HS-API the full national and area code must be provided.
9. If Insured Person Detail are included, either the Policy Number or the National Insurance Number must be completed by the user. The surname and forename are also mandatory if the Insured Person is not the patient who is the card holder.
10. Instances of Coded Clinical Details, Blood Group and Transfusion, Immunisation Details and Medication Details may be generated directly from Boolean representations of particular diseases etc or from more detailed information stored on the card. They may also exist and be displayed simply to indicate that the card does not support the particular data item or the particular data item is not recorded on a particular card. The “not recorded” indicator may also be used when the information is recorded but is in a secure area of the card which is not accessible to the interoperable application.
11. The “Clinical Emergency Category”, “Medication Emergency Category” and “Immunisation Emergency Category” values need not be stored explicitly on the card. They can be derived by mapping from:
    - tags such as those used in CardLink 1;
    - the emergency clinical data bitmap specified in ENV12018;
    - clinical codes used in accordance with ENV12018;
    - other forms of representation specific to particular card issuers.
12. The “Indicator” items need not be stored on the card but are derivable within the Healthcard Server. Furthermore, if the item is mandatory at the HS-API the Healthcard Server can generate appropriate information even if there is no relevant data on the card. For example, the absence of a particular item of information from a card that is able to store it is represented as 8 (not recorded). On the other hand, if a particular healthcard is unable to store the relevant information the Healthcard Server for that card can be programmed to return 9 (unsupported).

Table 14. Requirements for clinical data at different levels in an interoperable card system and a comparison with current data sets

Name	HS-API optionality	Card design optionality	Card user optionality	Application visibility	CardLink	DiabCard
Diseases	**For structure see coded clinical details**					
Asthma	M	M	O	M	ü	ü
Heart Disease	M	M	O	M	ü	ü
Cardiovascular Disease	O	R	O	R	û	ü
Epilepsy	M	M	O	M	ü fits	ü
Neurological disorders	O	R	O	R	û	ü neuropathy
Coagulation deficiency	M	M	O	M	ü	ü
Diabetes	M	M	O	M	ü	ü
Glaucoma	M	M	O	M	ü	ü
Other chronic diseases	O	O	O	O	ü ICD9.CM	ü free text
Procedures	**For structure see coded clinical details**					
Dialysis Treatment	M	M	O	M	ü	ü
Removal of an Organ	M	M	O	M	ü missing organ	ü
Transplanted Organ	M	M	O	M	ü	ü
Removable Prosthesis	M	M	O	M	ü	ü
Pacemaker	M	M	O	M	ü	ü
Other Procedures	O	O	O	O	ü important operation	ü important operation

Name	HS-API optionality	Card design optionality	Card user optionality	Application visibility	CardLink	DiabCard
Allergies	**For structure see coded clinical details**					
Analgesics	O	O	O	M	ü coded	ü
Animal hair	O	O	O	M	ü coded	ü
Antibiotics	O	O	O	M	ü coded	ü
Citrus fruits	O	O	O	M	ü coded	ü
Dust (or dust mite)	O	O	O	M	ü coded	ü
Eggs	O	O	O	M	ü coded	ü
Fish or Shellfish	O	O	O	M	ü coded	ü
Iodine	O	O	O	M	ü coded	ü
Milk	O	O	O	M	ü coded	ü
Nuts	O	O	O	M	ü coded	û
Pollen	O	O	O	M	ü coded	ü
Other Allergies	O	O	O	O	ü coded	ü free text

Name	HS-API optionality	Card design optionality	Card user optionality	Application visibility	CardLink	DiabCard
MEDICATION	**For structure see medication details**					
Anti-arrhythmic	M	M	O	M	ü	ü
Anti-coagulants	M	M	O	M	ü	ü
Anti-convulsants	M	M	O	M	ü	ü
Anti-diabetics	M	M	O	M	ü no distinction from insulin	ü
Anti-histamines	M	M	O	M	ü	ü
Anti-hypertensives	M	M	O	M	ü	ü
Beta blockers	M	M	O	M	ü	û
Corticosteroids	M	M	O	M	ü	û
Cytostatics & cytotoxics	M	M	O	M	ü Cytostatics	û
Digitalis	M	M	O	M	ü	û
Diuretics	M	M	O	M	ü	û
Insulin	M	M	O	M	ü unspecified anti-diabetics	ü
Monoamine oxidase inhibitors	M	M	O	M	ü	û
Psycholeptics	M	M	O	M	ü	û
Others	O	O	O	O	û	û
IMMUNISATIONS	**For structure see immunisation details**					
Anthrax	O	O	O	R	ü	û/ü text only
BCG	O	O	O	R	ü	û/ü text only
Cholera	O	O	O	R	ü	û/ü text only
Diphtheria	O	O	O	R	ü	û/ü text only
Diphtheria, Pertussis & Tetanus	O*	O*	O	M	ü	û/ü text only
Diphtheria & Tetanus	O*	O*	O	M	ü	û/ü text only
Haemophilus Influenza B	O	O	O	R	ü	û/ü text only

Name	HS-API optionality	Card design optionality	Card user optionality	Application visibility	CardLink	DiabCard
IMMUNISATIONS (continued)						
Hepatitis A	O	O	O	R	ü	û/ü text only
Hepatitis B	O	O	O	R	ü	û/ü text only
Influenza	O	O	O	R	ü	û/ü text only
Japanese encephalitis	O	O	O	R	ü	û/ü text only
Measles	O	O	O	R	ü	û/ü text only
Measles, Mumps and Rubella	O	O	O	R	ü	û/ü text only
Measles & Rubella	O	O	O	R	ü	û/ü text only
Meningococcal Infection (A&C)	O	O	O	R	ü	û/ü text only
Mumps	O	O	O	R	ü	û/ü text only
Pertussis	O	O	O	R	ü	û/ü text only
Pneumococcus	O	O	O	R	ü	û/ü text only
Polio (inactivated vaccine)	O	O	O	R	ü	û/ü text only
Polio (oral vaccine)	O	O	O	R	ü	û/ü text only
Rabies	O	O	O	R	ü	û/ü text only
Rubella	O	O	O	R	ü	û/ü text only
Tetanus	M	M	O	M	ü	ü
Tick Borne Encephalitis	O	O	O	R	ü	û/ü text only
Typhoid (oral)	O	O	O	R	ü	û/ü text only
Typhoid (injection)	O	O	O	R	ü	û/ü text only
Yellow Fever	O	O	O	R	ü	û/ü text only
Others	O	O	O	O	û	û/ü text only

8.4 ASN.1 Representation of API data structures

This section provides a formal ASN.1 representation of the data items as represented at the Healthcard Server API. More detailed notes on usage are given in Section 8.2 – Tables 6, 7 and 8.

8.4.1 Healthcard Server API Data

```
HealthcardServerApiData ::= SET
{
cardApplicationData             [0] CardApplicationData,
administrativeData              [1] AdministrativeData,
clinicalData                    [2] ClinicalData
}
```

8.4.2 Card Data

```
CardApplicationData ::= SET
{
cardIssuerIdentifier            [0] CardIssuerIdentifier,
cardHolderIdentifier            [1] OCTET STRING (SIZE (0...21)) OPTIONAL,
cardIdentifier                  [2] OCTET STRING (SIZE (0...28)),
cardStatus                      [3] ENUMERATED
                                {Unknown(0), Test(1), Normal(2)},
cardApplicationIdentification   [61] SET SIZE (1…9) OF ApplicationTemplate
}

CardIssuerIdentifier ::= SET
{
majorIndustryIdentifier         [0] NUMERIC STRING DEFAULT 80 (SIZE(2)),
countryCode                     [1] NUMERIC STRING (SIZE(3)),
issuerIdentifier                [2] NUMERIC STRING (SIZE(5…8)),
checkDigit                      [3] NUMERIC STRING (SIZE(1))
}

ApplicationTemplate ::= SET
{
cardApplicationIdentifier       [4F] OCTET STRING Size (0…16) ; as defined in ISO7816-5
discretionaryApplicationData    [73] DiscretionaryData

DiscretionaryData ::= SET
cardApplicationType             [0] ENUMERATED
                                {Administrative and Emergency Clinical(0), Administrative(1),
                                Emergency Clinical(2), Local use(9)},
cardApplicationVersion          [1] NUMERIC STRING (SIZE(2))
}
```

8.4.3 Administrative Data

```
AdministrativeData ::= SET
{
patientIdentification           [0] SET SIZE (1…3) OF PatientIdentification,
nameDetails                     [1] NameDetails,
languageDetails                 [2] SEQUENCE SIZE (0…4) OF LanguageDetails OPTIONAL,
birthDetails                    [3] BirthDetails,
addressDetails                  [4] SET SIZE (0…2) OF AddressDetails OPTIONAL,
contactDetails                  [5] SET SIZE (0…3) ContactDetails OPTIONAL,
insuringBodies                  [6] SET SIZE (0…3) OF InsuringBodyDetails OPTIONAL,
}
```

```
PatientIdentification ::= SET
{
issuerOfPatientIdentifier       [0] IssuerOfPatientIdentifier,
patientIdentifier               [1] OCTET STRING (SIZE (0...21))
}

NameDetails ::= SET
{
title                           [0] OCTET STRING (SIZE (0...7)) OPTIONAL,
surnamePrefix                   [1] OCTET STRING (SIZE (0...15)) OPTIONAL,
surname                         [2] OCTET STRING (SIZE (0...27)),
alternativeSurname              [3] OCTET STRING (SIZE (0...27)) OPTIONAL,
surnameSuffix                   [4] OCTET STRING (SIZE (0...15)) OPTIONAL,
forenames                       [5] SEQUENCE SIZE (1…3) OF OCTET STRING (SIZE (1...16)),
preferredForename               [6] OCTET STRING (SIZE (1...16)) OPTIONAL
surnameAtBirth                  [7] OCTET STRING (SIZE (0...27)) OPTIONAL,
}

LanguageDetails ::= SET
{
language                        [0] OCTET STRING (SIZE (2))
abilityInLanguage               [1] ENUMERATED
                                {preferred(0), fluent(1), fair(2), poor(3)} OPTIONAL,
}

BirthDetails ::= SET
{
dateOfBirth                     [0] Date,
sex                             [1] ENUMERATED
                                {unknown(0), male(1), female(2), other(3), other(9)}
countryOfBirth                  [2] NUMERIC STRING (SIZE (3)) OPTIONAL
}

AddressDetails ::= SET
{
addressStatus                   [0] ENUMERATED
                                {current home address(0), previous home address(1)}
addressStructure                [1] AddressStructure OPTIONAL,
telecomStructure                [2] TelecomStructure OPTIONAL
}

ContactDetails ::= SET
{
contactName                     [0] OCTET STRING (SIZE (0...30)),
contactRelationship             [1] OCTET STRING (SIZE (0...30)) OPTIONAL,
contactAddressStructure         [2] AddressStructure OPTIONAL,
contactTelecomStructure         [3] TelecomStructure OPTIONAL
}

InsuringBodyDetails ::= SET
{
insuringBodyCountry             [0] NUMERIC STRING (SIZE(3)) OPTIONAL,
insuringBodyIdentifier          [1] NUMERIC STRING (SIZE(9)),
insuringBodyName                [2] OCTET STRING (SIZE (0...20)) OPTIONAL,
insuringBodyAddressStructure    [3] AddressStructure OPTIONAL,
insuringBodyTelecomStructure    [4] TelecomStructure OPTIONAL,
e111Certificate                 [5] E111Certificate OPTIONAL,
insuranceNumbers                [6] InsuranceNumbers,
insuredPerson                   [7] InsuredPerson OPTIONAL
}
```

```
InsuranceNumbers ::= SET
{
insuredPersonPolicyNumber       [0] OCTET STRING (SIZE (0...21)) OPTIONAL,
nationalInsuranceNumber         [1] OCTET STRING (SIZE (0...21)) OPTIONAL,
}

InsuredPerson ::= SET
{
relationshipToPatient           [0] OCTET STRING (SIZE 0…16)) OPTIONAL,
insuredPersonSurname            [1] OCTET STRING (SIZE (0...27)),
insuredPersonAlternativeSurname [2] OCTET STRING (SIZE (0...27)) OPTIONAL,
insuredPersonForenames          [3] OCTET STRING (SIZE (0...16)),
insuredPersonAddressStructure   [4] AddressStructure OPTIONAL,
insuredPersonTelecomStructure   [5] TelecomStructure OPTIONAL
}

IssuerOfPatientIdentifier ::= SET
{
majorIndustryIdentifier         [0] NUMERIC STRING (SIZE(2)),
countryCode                     [1] NUMERIC STRING (SIZE(3)),
issuerIdentifier                [2] NUMERIC STRING (SIZE(5…8)),
checkDigit                      [3] NUMERIC STRING (SIZE(1))
}

AddressStructure ::= SET
{
addressText                     [0] SEQUENCE SIZE (1…5) OF OCTET STRING (SIZE (0…35))
addressPostcode                 [1] OCTET STRING (SIZE (0...8)) OPTIONAL,
addressCountry                  [2] NUMERIC STRING (SIZE (3)) OPTIONAL
}

TelecomStructure ::= SET
{
telephoneNumber                 [0] SEQUENCE SIZE (0…3) OF NUMERIC STRING (SIZE(0…16))
                                    OPTIONAL,
facsimileNumber                 [1] NUMERIC STRING (SIZE(0…16)) OPTIONAL,
networkAddress                  [2] OCTET STRING (SIZE (0...32)) OPTIONAL
}

E111Certificate ::= SET
{
expiryDate                      [0] Date
}
```

8.4.4 Clinical Data

```
clinicalData ::= SET
{
codedClinicalDetails            [0] SET SIZE (1…99) OF CodedClinicalDetails,
bloodGroupTransfusionDetails    [1] BloodGroupTransfusionDetails,
immunisationDetails             [2] SET SIZE (0…10) OF ImmunisationDetails OPTIONAL,
medicationDetails               [3] SET SIZE (1…30) OF MedicationDetails,
clinicalAddressDetails          [4] SET SIZE (0…9) OF ClinicalAddressDetails OPTIONAL,
opticalPrescriptionDetails      [5] OpticalPrescriptionDetails OPTIONAL,
updateDetails                   [6] UpdateDetails
}
```

CodedClinicalDetails ::= SET
{
clinicalEmergencyCategory [0] NUMERIC STRING (SIZE(2)),
clinicalIndicator [1] ENUMERATED
{Absent(0), Present(1), Possible(2), Not recorded(8), Not supported(9)},
clinicalCodingStructure [2] ClinicalCodingStructure OPTIONAL,
clinicalDate [3] Date OPTIONAL,
clinicalText [4] OCTET STRING (SIZE (0...80)) OPTIONAL,
clinicalEntryDate [5] Date OPTIONAL,
clinicalAuthor [6] Author OPTIONAL
}

ClinicalCodingStructure ::= SET
{
codingSchemeIdentifier [0] OCTET STRING (SIZE (6)),
clinicalCode [1] OCTET STRING (SIZE (0…8))
}

BloodGroupTransfusionDetails ::= SET
{
bloodGroup [0] BloodGroup,
bloodTransfusion [1] BloodTransfusion,
bloodGroupEntryDate [2] Date OPTIONAL,
bloodGroupAuthor [3] Author OPTIONAL
}

BloodGroup ::= SET
{
aBOBloodGroup [0] OCTET STRING (SIZE (1…2)),
rhesusFactor [1] OCTET STRING (SIZE (1)),
dateOfLastBloodGrouping [2] Date OPTIONAL,
bloodGroupingText [3] OCTET STRING (SIZE (0...30)) OPTIONAL
}

BloodTransfusion ::= SET
{
bloodTransfusionIndicator [0] ENUMERATED
{Never(0), One or more (1), Unknown(2), Not recorded(8), Not supported(9)},
lastBloodTransfusionDate [1] Date OPTIONAL
}

ImmunisationDetails ::= SET
{
immunisationEmergencyCategory [0] NUMERIC STRING (SIZE(2)),
immunisationIndicator [1] ENUMERATED
{Never(0), One or more(1), Unknown(2), Adverse reaction(4), Not recorded(8), Not supported(9)},
immunisationStatus [2] ENUMERATED
{Unspecified(0), First dose(1), Second dose(2), Third dose(3), Completed course(4), Booster(5), Not supported(9)},
lastDateImmunised [3] Date OPTIONAL,
immunisationCodingStructure [4] ClinicalCodingStructure OPTIONAL,
immunisationText [5] OCTET STRING (SIZE (0...30)) OPTIONAL
immunisationEntryDate [6] Date OPTIONAL,
immunisationAuthor [7] Author OPTIONAL
}

MedicationDetails ::= SET
{

medicationEmergencyCategory	[0] NUMERIC STRING (SIZE(2)),
medicationIndicator	[1] ENUMERATED {Absent(0), One or more(1), Unknown(2), Past or short term(5), Intermittent(6), Not recorded(8), Not supported(9)},
medicationCodingStructure	[2] SET SIZE (0…6) OF ClinicalCodingStructure OPTIONAL,
medicationDrugName	[3] OCTET STRING (SIZE (0...50)) OPTIONAL,
medicationDosageCode	[4] SET SIZE (0…4) OF OCTET STRING (SIZE (0...2)) OPTIONAL,
medicationDosage	[5] OCTET STRING (SIZE (0...50)) OPTIONAL,
medicationStartedDate	[6] Date OPTIONAL,
medicationEndedDate	[7] Date OPTIONAL
medicationEntryDate	[8] Date OPTIONAL,
medicationAuthor	[9] Author OPTIONAL

}

ClinicalAddressDetails ::= SET
{

clinicalAddressName	[0] OCTET STRING (SIZE (0...30)),
clinicalAddressRelationship	[1] OCTET STRING (SIZE (0...16)) OPTIONAL,
clinicalAddressStructure	[2] AddressStructure OPTIONAL,
clinicalTelecomStructure	[3] TelecomStructure OPTIONAL

}

OpticalPrescriptionDetails ::= SET
{

opticalPrescription	[0] OCTET STRING (SIZE (0...40)),
opticalPrescriptionDate	[1] Date OPTIONAL

}

UpdateDetails ::= SET
{

dateOfLastClinicalUpdate	[0] Date,
responsibleParty	[1] Author OPTIONAL

}

Author ::= SET
{

authorCountry	[0] NUMERIC STRING (SIZE(3)) OPTIONAL,
authorIdentifier	[1] NUMERIC STRING (SIZE(0…12)) OPTIONAL,
authorName	[2] OCTET STRING (SIZE (0...20)) OPTIONAL

}

9 Annexes

Annex A. Standards relevant to interoperable healthcards

International Standards

ISO 639	Codes for the representation of names of languages.
ISO 646	Information Technology - ISO 7 -bit coded character set for information interchange.
ISO 2014	Representation of calendar dates in an all numeric form.
ISO 2015	Numbering of weeks.
ISO 2955	Information processing - Representation of SI and other unit in systems with limited character sets.
ISO 3307	Representation of time of day.
ISO 4031	Representation of local time differentials.
ISO 4217	Codes for the representation of currencies and funds.
ISO 5218	Information interchange - Representation of human sexes.
ISO 6093	Information processing - Representation of numerical values in character strings for information interchange.
ISO 6523	Data Interchange - Structures for the identification of organisations.
ISO 6709	Standard representation of latitude, longitude and altitude for geographic point location.
ISO 7498-2	Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2 : Security Architecture (1989-02-15).
ISO 7501-1	Identification cards - Machine readable Travel documents :1993 - Part 1: Machine readable passport. - Part 2: Machine readable visa.
ISO 7810	Identification cards - Physical characteristics.
ISO 7811-1	Identification cards - Recording technique - Part 1: Embossing.
ISO 7811-2	Identification cards - Recording technique - Part 2: Magnetic stripe.
ISO 7811-3	Identification cards - Recording technique - Part 3: Location of embossed characters on ID1-cards.
ISO 7811-4	Identification cards - Recording technique - Part 4: Location of read-only magnetic tracks. Tracks 1 and 2.
ISO 7811-5	Identification cards - Recording technique - Part 5: Location of read-only magnetic track. Track 3.
ISO 7812	Identification cards - Numbering system and registration procedure for issuer identifiers.
ISO 7811	Embossing and Magnetic stripes
ISO 7812	Identification of issuers and applications
ISO 7813	Financial transaction cards

International Standards (continued)

ISO 7816-1	Integrated circuit(s) cards with contacts - Physical characteristics
ISO 7816-2	Integrated circuit(s) cards with contacts - Dimensions and location of the contacts
ISO 7816-3	Integrated circuit(s) cards with contacts - Electronic signals and transmission protocols
ISO 7816-4	Integrated circuit(s) cards with contacts - Inter-industry commands for interchange
ISO 7816-5	Integrated circuit(s) cards with contacts - Numbering system and registration for applications
ISO 7816-6	Integrated circuit(s) cards with contacts - Inter-industry data elements
ISO 7826	Information interchange - Registration and identification of coding schemes.
ISO 8601	Representation of dates and times.
ISO 8824	Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) (Version 2 1991-04-24).
ISO 8825	Information technology - Open Systems Interconnection Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). (Version 2 1991-04-24).
ISO 8859-1	Information processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1. :1987
ISO 9594-4	Information technology - Open Systems interconnection : The Directory - Part 8 : Authentication framework.
ISO 9798	Information technology - Security techniques - Entity authentication mechanisms - Part 1 : General model.
ISO 10373	Identification cards - Test methods
ISO 10181-2	Information technology - Open systems interconnection - Security frameworks for Open Systems : Authentication Framework.
ISO 10202-1	Financial transaction cards - Security architecture of financial transaction systems using integrated circuit cards.

CCITT Standards

CCITT	Numbering plan for the international telephone service Recommendation E.163
CCITT X.509	The Directory - Authentication framework Recommendation

European Standards and Prestandards

EN 742	Identification cards - ID-1 card location of contacts for cards and devices used in Europe
EN 1068	Medical informatics - Healthcare information interchange- Registration of coding schemes.
EN 1387	Identification card systems - Health care applications - Cards : General characteristics.
EN 3166	Codes for the representation of countries (Third edition 1988-08- 15).
ENV 12018	Identification, administrative and common clinical data structures for intermittently connected devices used in health care(including machine readable cards).
EN 27 810	Identification cards - Physical characteristics.(see ISO 7810)
EN 27 811:1989	Identification cards - Recording technique - (see ISO 7811)
EN 27 812:1989	Identification cards - Numbering system and registration procedure for issuer identifiers. (see ISO 7812)
EN 27 816:1989	Identification cards - Integrated circuit(s) with contacts - (see ISO 7816)

European Standards in development

prEN1867	Machine readable cards - Healthcare applications - Numbering systems and registration procedure for issuer identifiers.
prEN1105	Identification card systems - General concepts applying to inter sector environments - Rules for inter-applications consistency.
Unnumbered:- (Active work item in CEN TC224 WG12)	Machine readable cards - Healthcare applications - Logical data structures and concepts for different card technologies for use by patients in health applications. *Note:* This builds technology dependent logical data structures based on the technology independent European Prestandard ENV12018.

Annex B. Validation of dosage codes

Table 15 shows the mapping from the EDBS[7] dosage coding (used by several UK GP systems) to the proposed healthcard interoperability dosage coding scheme based on the CardLink 1 EHB proposals (see Table 12).

Up to four EHB codes can be combined to produce more complex sets of instructions.

The EDBS codes are paired so that each dosage instruction consists of one code from the quantity set and one code from the timing set.

Results of the validation exercise

- The EHB dosage codes are more extensive and cover a wider range of functionality including route of administration and warnings which the EDBS system codes differently (not shown here).
- There are six EDBS codes that have no obvious equivalent in EHB.
- There are also five EDBS codes that are only partially matched in EHB.
- Both sets of codes appear to include some duplications where different codes have the same text associated with them (marked * in the table below).
- The EHB codes have higher incidence of duplication as several of codes can also be represented by combinations of elements.

Table 15. Dosage code validation

EDBS quantity codes		Equivalent EHB codes	
1	take one	T1	.take one
2	take two	T2	.take two
3	take three	T3	.take three
4	take four	T4	.take four
5	20ml	20	.take four 5ml spoonfuls
6	take 1 or 2,	12	.one or two to be taken
7	apply	AP	.apply
8	one drop	D1	.instil one drop
9	two drops	D2	.instil two drops
10	15ml	15	.take three 5ml spoonfuls
11	half tablet	HT	.half a tablet
12	insert one	I1	.insert one
13	insert two	I2	.insert two
14	2.5ml	.5	.take half a 5ml spoonful
15	1ml		**NO MATCH**
16	one puff	1P	.one puff to be inhaled
17	two puffs	2P	.two puffs to be inhaled
18	1x5ml spoon	5* 05*	.take one 5ml spoonful .take one 5ml spoonful
19	2x5ml spoon	10	.take two 5ml spoonfuls

[7] This scheme is supported by Exeter Data Base Systems (EDBS) in their drug database systems. These are used by several significant UK GP system suppliers. The version shown here is based on 1993 documentation that may now be out of date and ins NOT intended for implementation. It is included only to illustrate the coverage of the EHB proposals and the comparison is not intended to imply merit of either system.

EDBS quantity codes		**Equivalent EHB codes**	
20*	use	US	.use
21	inject one	IJ	**PARTIAL MATCH** .for injection only
22	decreasing dosage	-1	**PARTIAL MATCH** then reduce by one each day until the course is complete
23	suck	SU	**PARTIAL MATCH** .to be sucked or chewed
24	chew	SU	**PARTIAL MATCH** .to be sucked or chewed
25	dissolve	DS	.dissolve
26*	use	US	.use
EDBS timing codes		**Equivalent EHB codes**	
1	daily	DA	daily
2	twice daily	BD	twice daily
3	3 times/day	TD	three times daily
4	4 times/day	QD	four times daily
5	5 times/day		**NO MATCH**
6	every 4 hours	4H* QH*	every four hours every four hours
7	every 3 hours	3H	every three hours
8	every 2 hours	2H	every two hours
9	hourly	1H	every hour
10	morn,1 night	MA+1N	in the morning + one at night
11*	as directed	MD	as directed
12	every 8 hours	8H	every eight hours
13	each morning	MA	in the morning
14	at night	NO	at night
15	as needed	PR	as required
16	day 12-26cyc		**NO MATCH**
17	every 6 hours	6H	every six hours
18	every 12 hours		**NO MATCH**
19	weekly		**NO MATCH**
20	<blank>		<blank>
21	morn,2 night	MA+2N	in the morning + two at night
22*	as directed	MD	as directed
23	twice weekly		**NO MATCH**
24	early evening	EV	**PARTIAL MATCH** in the evening

Annex C Mapping between CTM-API and CardLink 1

This annex illustrates the mapping from the CTM-API functions defined in Section 5 to the relevant CardLink 1 card reader interface functions.

The intention is to illustrate the possibility for simple mapping and to assist consistent interpretation. It is not intended that this annex should be used as a definitive technical specification.

CtmOpen : Open session with CT-Manager

CtmOpen(pAppHdl)

No equivalent in CardLink. This is a new function required to support Card Terminal management.

Close session with CT-Manager

CtmClose(AppHdl)

Map to CardLink 1 interface function:

Close_Reader(*TermIdent*)

This should be applied to all *ResHdl* entries allocated to the calling application.

Open logical link with a Card Terminal resource

UINT16RC CtmResOpen(AppHdl, szResName, pResHdl)

Map to CardLink 1 interface function:

OpenReader (*LOGLineNam*, *TermIdent*)

LOGLineName is a name that related to the Card Terminal connection. This should be mappable from information in the CTM configuration file entries indicated by *szResName*.

TermIdent is a structure including *Connex*_SC and this should map to the logical handle *ResHdl* for future communications with the relevant card.

In the case of a card reader with two slots *Connex_AUX1* refers to the second slot. According to the configuration some requests may need to be mapped to *Connex_AUX1*. In this case the relevant information can be stored in the CTM configuration file so that the mapping can be handled as required.

Subsequent CTM-API commands that reference *ResHdl* will either use this directly as the CardLink *Connex* number or map it according to the configuration file.

Set exclusivity

CtmSetExclusivity(ResHdl, ExclScope)

No equivalent in CardLink. This is a new function required to support Card Terminal management.

Request or check card insertion

CtmCardOpen (ResHdl, SzPrompt, EffectFlags, TimeOut, plenATR, pATR, pCardState, pStatus)

Map to CardLink 1 sequence of interface functions:

Reader_Status(*Connex, Status*)

Map *ResHdl* to the relevant *Connex* (as described under *CtmResOpen*).

The status is used to determine if a card is not inserted. The command is repeated until the *TimeOut* expires or a card is inserted. When a card is inserted the **NewCard** function can be called to power on and reset the card.

NewCard(p_*data*, *Connex*)

Map *ResHdl* to the relevant *Connex* (as described under *CtmResOpen*).

If necessary repeat until the *TimeOut has expired*.

The *pStatus* and *pCardState* parameters are set according to the end result of the sequence.

p_*data* includes the ATR that needs to be returned in the *CtmCardOpen* buffer.

Power-off and enable card removal

CtmCardClose(ResHdl, SzPrompt, EffectFlags, TimeOut, pCardState, pStatus)

Map to CardLink 1 interface function:

RemoveCard(*TimeOut, Connex)*

Map *ResHdl* to the relevant *Connex* (as described under *CtmResOpen*).

TimeOut is used directly.

The *pStatus* and *pCardState* parameters are set according to the end result of the function.

Test the state of a card or ICC slot

CtmCardTest(ResHdl, pCardState, pStatus)

Map to CardLink 1 interface function:

Reader_Status(*Connex, Status*)

Map *ResHdl* to the relevant *Connex* (as described under *CtmResOpen*).

The *pStatus* and *pCardState* parameters are set according to the Status reported.

Transparent command exchange with card

CtmCardCommand(ResHdl, lenCommand, pCommand, plenResponse, pResponse, pCardState, pStatus)

Map to CardLink 1 sequence of interface functions:

InputCard(*Connex*, *Cmd_card*, *TimeOut*)

Map *ResHdl* to the relevant *Connex* (as described under *CtmResOpen*).

Cmd_card has the same form as *pCommand*.

TimeOut can be set as a Card Terminal configuration option.

After calling *InputCard* the

Get_Output(*p_stat*, *p_data*, *Connex*)

Map *ResHdl* to the relevant *Connex* (as described under *CtmResOpen*).

p_stat contains the card status indicating card level errors in command execution.

p_data contains the response from the card.

The combination of p_data and p_stat are returned as pResponse.

The *pStatus* and *pCardState* parameters are set according to the end result of the sequence.